

C程序设计

E2



在线评判系统

厦门大学信息学院软件工程系

黄炜 副教授

在线测评系统

• 网址

- `http://whuang.spimag.com:20101`
- `http://whuang.imwork.net:20101`



导语

- OJ 君是一个陪练，用来陪你锻炼编程能力，掌握以下技能：
 - 从问题到解法，再将解法抽象成一个规律
 - 再优化这个计算方法
 - 最后用代码形式表达出来，告诉电脑如何完成。
- 因此，掌握技能为先，答对多少题是次要的。



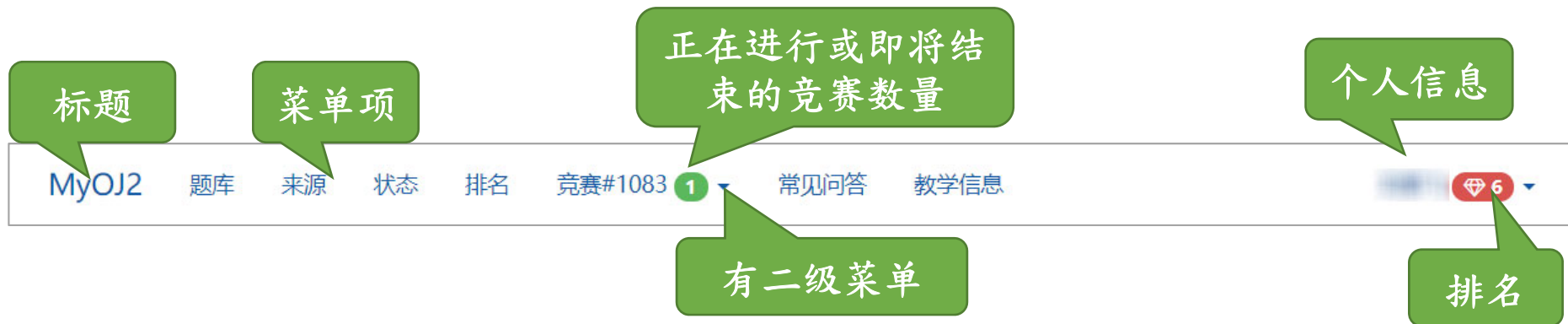
界面

• 主页



菜单

• 菜单栏



登录页面

- 用户名和密码唯一标识一个用户。如果验证码看不清，就单击一下图片，换个验证码。



习题集

- 根据题号、积分难度或关键字找到题目。

根据题号查题

按类别查看题目

根据关键字搜题

题号 查找

来源 分类 错误

标题或来源的关键词 查找

我做对了这题

1 2 3 4 5 6 7 8 9 10 11

答对用户越多，
积分越低

翻页

标题，点进去看题

已解

题号 ^

标题

提交

答对用户

最佳用户

积分



1000

A+B Problem

测试题

答对
人次

370

最佳用户

1.85

1001

Solution of a second order equation

Wei Huang

140

最佳用户

2.58



1002

Solution of a real second order equation

Wei Huang

451

167

最佳用户

2.34

1005

Cycle, Sphere and Cylindrical Calculation

Wei Huang

425

121

最佳用户

2.76

1006

Integer addition

Wei Huang

提交
次数

143

最佳用户

运行时间最
短的用户

题目来源

我做了这题，
但还没答对

题号



2019-10-09

厦门大学信息学院软件工程系 黄炜

题目页

The screenshot shows a web browser window with the address bar displaying 'whuang.imwork.net:20101/OJ/problem-1000.html'. The page title is 'MyOJ2: 题目 1000: A+B Problem'. The main content area has a blue header with the title 'A+B Problem' and the problem number '题目 1000'. Below the header, there are sections for '描述' (Description), '输入' (Input), and '输出' (Output). The description says 'Calculate $a + b$ '. The input section says 'Two integers a, b ($0 \leq a, b \leq 10$) in a single line.' The output section says 'Output $a + b$ in a single line.'

Annotations (green boxes with arrows):

- 标题 (Title) points to 'A+B Problem'.
- 题号 (Problem Number) points to '题目 1000'.
- 题目描述 (Problem Description) points to 'Calculate $a + b$ '.
- 这里说明你的程序需要符合的一类测试数据，即：测试数据只有这个类型的。 (Here it indicates a class of test data your program needs to conform to, i.e., test data is only of this type.) points to the input section.
- 注意类型和取值范围，可以简化问题 (Pay attention to the type and value range, which can simplify the problem) points to the input section.
- 这里说明你的程序的输出应该符合这个标准。 (Here it indicates that your program's output should conform to this standard.) points to the output section.



题目页

这里提供了一个简单的示例，旨在帮助你理解输入的格式，真正的测试数据更多更坑！

这里用蓝色背景表示输入数据的起止，这表示1、空格、2，并且之后没有空格。这是等宽字体，个别题目输出行的后面还有蓝色背景的空白，则表示有空格。

这里提供了一个在样例输入下的输出示例，旨在帮助你理解输出的格式。

内存最大占用不能超过10MB

本题共608人次提交，其中370人次答对，答对将得到1.8499分。

后台数据测试时，总时间不能超过1秒

时间限制: 1 秒; 内存限制: 10 MB; 提交: 608; 已解决: 370; 积分: 1.8499.

永久性连接: <http://whuang.imwork.net:20101/OJ/problem-e3cbba8883fe746c6e35783c9404b4bc0c7ee9eb.html>

提示

关于解题的一点提示

请仔细阅读 [常见问题](#) 页面，并且提交代码测试你的系统和浏览器是否正确运行。

Please read the [FAQ page](#) carefully and submit the code to test whether your system and browser may work correctly.

通过完成本题，掌握：



题目页

MyOJ2: 题目 1000: A+B Proble x

① 不安全 | whuang.imwork.net:20101/OJ/problem-1000.html

9. 应测试【样例输入】，并确保得到【样例输出】没有任何一个字符的错误；
10. 应根据【输入】的描述，猜测系统背后可能使用的测试用例输入和输出，并逐一分类测试无误；
11. 所有一切无误之后，方能提交。

来源

测试题

标签

测试题

提交 状态 讨论 0 加入收藏

我要提交

我想看看谁答对过

加入收藏夹

如果你有合适的解题报告或意见发此处。

题目出处，单击此处可以找到相同出处的题目

题目标记，注释。

MyOJ2 在线测评系统

@2014-2019 黄炜 保留其自行研发部分的所有权利。项目基于 HUSTOJ 项目组 在 2015年7月10日 的版本增强。部分题目镜像于厦门大学软件学社 OJ。服务器与域名受矽图（厦门）科技有限公司资助。
备案号：闽ICP备17023897号-1

② 获取帮助

常见问题
用户手册
致谢
反馈

语言

简体中文
正體中文
English

链接

图像码项目
厦大软件学社OJ
洛谷 OJ
杭电ACM系统



提交页

The screenshot shows the MyOJ2 submission interface. The left sidebar contains the text 'MyOJ2' and '提交 题目 100'. The main area has a '语言' (Language) dropdown set to 'C' and a '源代码' (Source Code) text area containing a C program. Below the code area is a '切换编辑器' (Toggle Editor) button. At the bottom, there are '样例输入' (Sample Input) and '样例输出' (Sample Output) fields, and a row of buttons: '提交' (Submit), '测试' (Test), and '状态' (Status).

题号

提交 题目 100

语言 C

源代码

```
1 #include <stdio.h>
2 int main(){
3     int a,b;
4     while(scanf("%d %d",&a, &b) != EOF)
5         printf("%d\n",a+b);
6     return 0;
7 }
```

选择编程语言 (也只有C了)

把你的答案贴到这里

文本编辑器和语法编辑器之间的转换

这里以样例输入和输出为默认值，仅在【测试】模式下用。

一般输入程序后应该单击【提交】，然后可以查看结果。

程序在系统得到的结果让你很困惑的时候使用。单击后数秒，样例输出栏会得到结果。

提交 测试 状态



状态页

The screenshot shows the 'MyOJ2: 状态' (MyOJ2: Status) page. The browser address bar shows 'whuang.imwork.net:20101/OJ/status.html'. The page has a navigation bar with links: '题库' (Question Bank), '文章' (Articles), '状态' (Status), '排名' (Ranking), '竞赛' (Contests), '常见问题' (FAQ), and '课程' (Courses). The user '黄炜' (Huang Wei) is logged in with a score of 61. The main heading is '状态' (Status). Below it, there are filters for '题目 ID' (Problem ID), '用户' (User), '语言' (Language), '结果' (Result), and '相似度' (Similarity). There are buttons for '查找' (Search), '我的状态' (My Status), and '运行数据' (Run Data). The page displays a list of submission records. Each record shows a status icon (e.g., '等待' (Waiting), '答错 9%' (Wrong 9%), '正确' (Correct)), a problem ID (e.g., 53388, 53387, 53386), a score (e.g., 1960, 1489, 1039), and various execution details like time (0.0s), memory (1.1M), and code length (0.3K). Annotations in green speech bubbles point to specific features: '根据题号、结果筛选' (Filter by problem number, result) points to the filter buttons; '只看我的提交' (Only view my submissions) points to the '我的状态' button; '翻页栏' (Page bar) points to the '前一页' and '后一页' buttons; '运行编号' (Run number) points to the problem ID; '结果图标' (Result icon) points to the status icons; '判题报告, 源码下载, 重新编辑提交' (Judgment report, source code download, re-submit) points to the links below a submission; '不停转圈说明后台卡住了' (Constant spinning indicates the backend is stuck) points to the '等待' status; '后台多组答案有几组是对的' (Backend has multiple answers, some are correct) points to the '答错 9%' status; '与以往答案相似度' (Similarity to previous answers) points to the '相似度' filter; and '运行内存, 时间, 语言, 和代码长度' (Run memory, time, language, and code length) points to the execution details.

MyOJ2: 状态

不安全 | whuang.imwork.net:20101/OJ/status.html

MyOJ2 题库 文章 状态 排名 竞赛 常见问题 课程

黄炜 61

状态

根据题号、结果筛选

只看我的提交

翻页栏

运行编号

结果图标

判题报告, 源码下载, 重新编辑提交

不停转圈说明后台卡住了

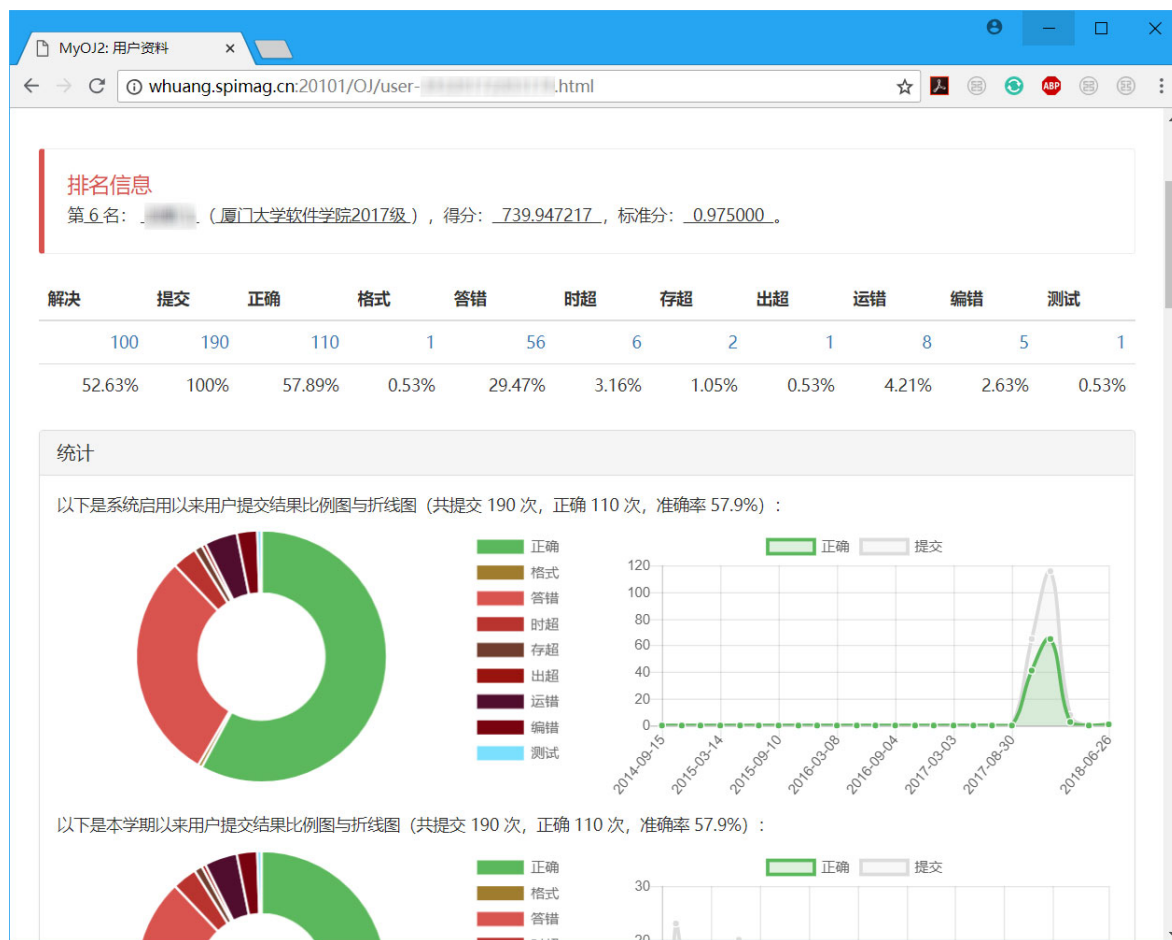
后台多组答案有几组是对的

与以往答案相似度

运行内存, 时间, 语言, 和代码长度



个人页



竞赛页

MyOJ2: 竞赛

用于本地时钟校准

竞赛

2018-12-30 19:58:33

链接到名校联赛

黄色框表示尚未开始的竞赛

绿色框表示正在进行的竞赛

蓝色框表示已结束的竞赛

单击竞赛标题进入

竞赛编号

1	2	3	4	5	6	名校联赛
 1109	 1108	 1107	 1106	 1105	 1104	
2018级期末上机测试	C积分赛 (2018年 决赛)	2018级算法训练 (12)	2018级作业 (12)	C积分赛 (2018年, 周赛 2)	2018级算法训练 (11)	
2019-01-02 10:10 - 2019-01-02 11:50	2018-12-30 20:00 - 2018-12-31 20:00	2018-12-28 20:00 - 2018-12-31 20:00	2018-12-26 20:00 - 2018-12-31 20:00	2018-12-22 20:00 - 2018-12-25 20:00	2018-12-21 20:00 - 2018-12-25 20:00	
排名 OI 排名 统计	题库 状态	题库 状态	题库 状态 排名 OI 排名 统计	题库 状态 排名 OI 排名 统计	题库 状态 排名 OI 排名 统计	



竞赛页

竞赛 1083

测试竞赛

开始时间

2018-06-

进度条

结束时间

2018-06-23 01:00:00

状态

运行中

公开

用于本地时钟校准

🕒 当前时间: 2018-06-21 23:19:12

状态

名次

名次 OI

统计

菜单栏

题目

标题

来源

正确

提交

题目 A

A+B Problem

用于测试

题目 B

Solution of a second order equation

数学

题目区



竞赛排名表

竞赛 1060

2017级C语言第4次作业 (A. 提高题)

查看截止到某个时间点
时的名次

2018-06-21 23:22:35

查找

导出

闪电标记说明
该用户在该题
做的最快

名次	用户	昵称	解决	罚分	[A]	[B]	[C]	[D]	[E]	[F]
1			6	65:33:27	13:15:59	01:30:04 (-4) ⚡	13:06:51 (-1)	18:18:50 (-5) ⚡	01:42:17 (-1) ⚡	13:19:26 (-2)
2			5	13:42:14	01:40:44	0	04:23:38	(-2)	45	01:25:35
3			5	35:41:1						
4			5	75:56:13	13:28:17	21:07:54 (-2)	23:53:34	(-2)	12:55:15 (-1)	03:31:13
5			4	20:11:29	01:08:23 ⚡	14:02:41 (-5)	(-6)	(-1)	11:24	01:09:01 ⚡
6			4	216:00:10	24:56:56 (-1)	75:24:23 (-2)				23:29:12

开赛的时间总和
加上每次提交错
误的罚分

负号说明该用户
提交出错次数

该用户提交正确
时离开赛的时间

红色底色说明该
用户提交未正确



排名

MyOJ2: 排名

whuang.imwork.net:20101/OJ/ranklist.html

排名

查找用户

按群组查看排名

按时间段查看排名

用户 查找

群组 查找 近期

日 周 月 学期 年

1-60 61-120 121-180 181-240 241-300 301-329

名次	姓名	学校	已解决	提交	准确率	积分	报表
1		厦门大学软件学院2018级	256	556	0.460	1738.23	
2		厦门大学软件学院2018级	260	448	0.580	1564.63	
3		厦门大学软件学院2018级	288	419	0.687	1463.78	
4		厦门大学软件学院2017级	267	574	0.465	1257.98	
5		厦门大学软件学院2018级	260	437	0.595	1155.45	
6		厦门大学软件学院2018级	241	363	0.664	1067.19	
7		厦门大学软件学院2018级	241	375	0.643	1016.40	
8		厦门大学软件学院2018级	187	430	0.435	863.94	



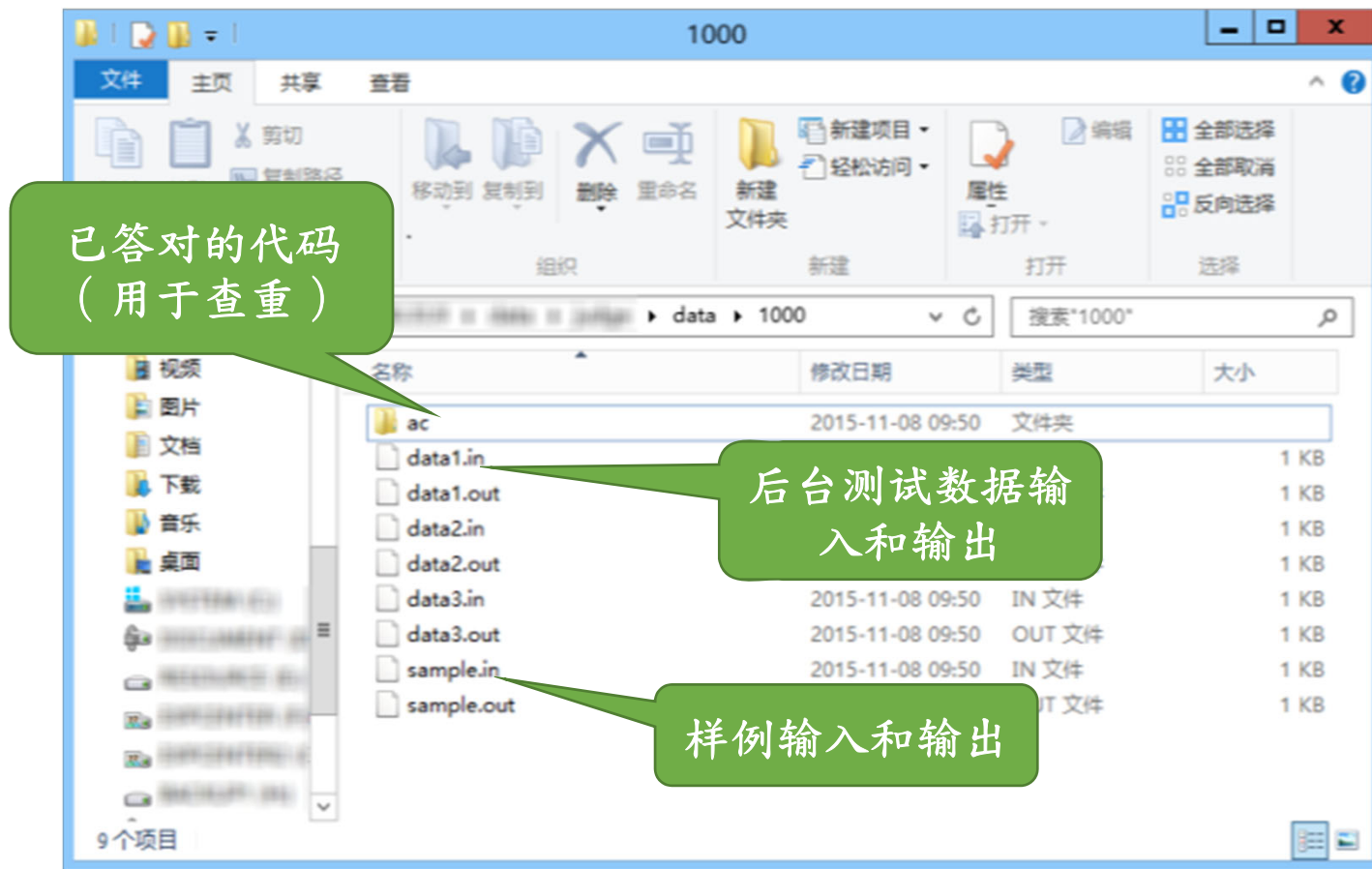
其它功能

- 收藏夹
- 短消息（日志）
- 管理面板
- 课程资料和作业



后台测试数据

- 后台测试数据文件夹结构



后台测试数据

- 测试数据并不限于“问题”页所显示的“样例输入”
- 用户提交前有义务调试自己的程序，多考虑极端情况。

示例输入： sample.in	示例输出： sample.out
1 2	3

测试数据1：输入 data1.in	测试数据1：输出 data1.out
1 3	4
3 1	4
9 9	18

测试数据2：输入 data2.in	测试数据2：输出 data2.out
0 3	3
1 1	2
9 1	10
4 5	9



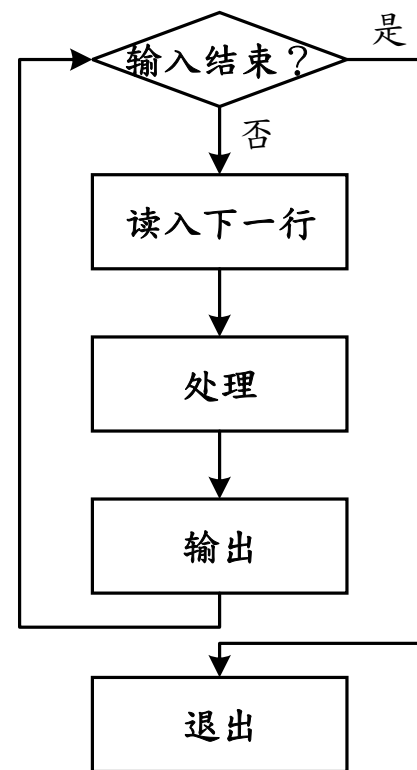
后台测试数据

• 循环输入

```
while (scanf("%d %d", &a, &b) != EOF)
```

#define EOF -1

- scanf返回EOF (-1) 表示输入结束
- 输入结束之前，重复运行，每行执行1次。
- 这样能配合有多个输入、重复次数未知、直至EOF结束的类型。
- 如果没有特殊说明，OJ都是这个类型。



后台测试数据

- 测试数据一般分成小数据、大数据和极限数据三种。
 - 小数据易于调试、易于设计、覆盖面广；
 - 大数据数据量比小数据大，同时可以使用较弱的替代算法得到结果的数据；
 - 极限数据是非常重要的一种数据，用来测试你的数组有没有越界。



程序判断流程

- 编译程序

- 编译不通过，则提示 编译错误

- 依次载入测试数据，直到所有数据测试完毕

- 运行程序

- 程序突然终止，提示 运行错误

- 监测占用内存变化，如果超过“限制内存”，提示 内存超限

- 等待规定的“限制时间”之后

- 程序仍未终止，提示 时间超限，并强行终止



程序判断流程

- 依次载入测试数据，直到所有数据测试完毕（续）
 - 得到程序在测试数据给定输入下的实际输出，与测试数据的参考输出做对比
 - 如果完全一致，则重复循环，测试下一组数据
 - 如果与参考输出相差若干个空白字符（空格、制表符、回车等）则提示 **格式错误**
 - 如果超过参考输出两倍长，则提示 **输出超限**
 - 否则，输出 **答案错误**
 - 所有数据测试完成，均和参考输出一致，则输出 **正确**



程序判断流程

- 如果系统打开支持代码查重
 - 与系统已有代码相似度超过50%？
 - 如果超过50%，输出 正确，相似度 xxx
 - 如果不超过 50% 输出 正确



作答之前的准备工作

- 准备编译环境

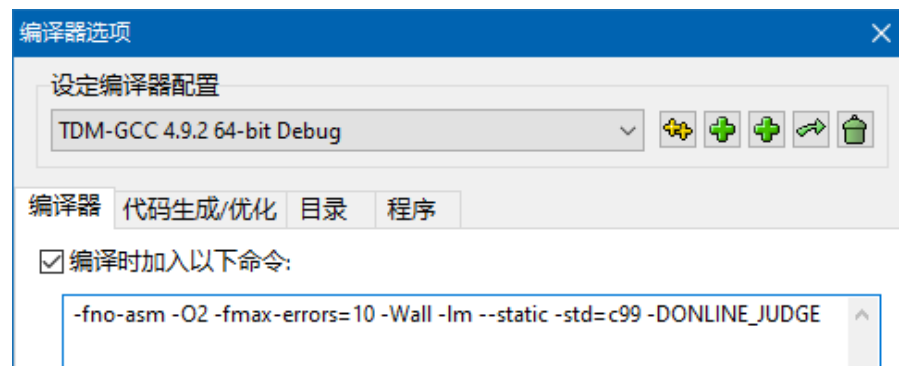
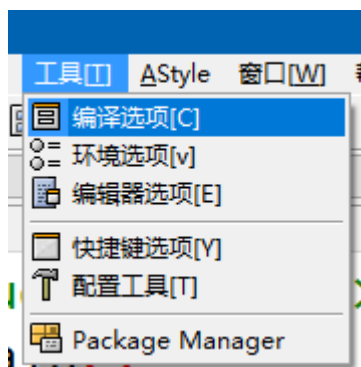
- 推荐使用DevC++ 5.1作为编译环境

- 根据FAQ页面设置编译选项

禁用汇编语言，打开2级优化，查出10个编译错误时停止，打开全部告警，链接数学库，静态编译，使用C99标准，使用 ONLINE_JUDGE 宏

语言	编译器	编译器选项 (假设文件名为 Main)
C	GNU GCC version 7.4.0	<code>gcc Main.c -o Main -fno-asm -O2 -fmax-errors=10 -Wall -lm --static -std=c99 -DONLINE_JUDGE</code>

- 错误的编译选项可能导致结果不一致



作答步骤

- 仔细审题

- 理解题意，了解输入输出
- 通过示例输入输出确认理解无误

- 先在自己的机器上输入程序。

用到了标准输入
输出函数scanf

可执行程序入口

程序多行，每行一
个测试实例，不能
有分号

```
#include <stdio.h>
int main() {
    int a,b;
    while (scanf("%d %d",&a, &b) != EOF)
        printf("%d\n",a-b);
    return 0;
}
```

声明语句
假设不慎输错标点

返回0，报平安

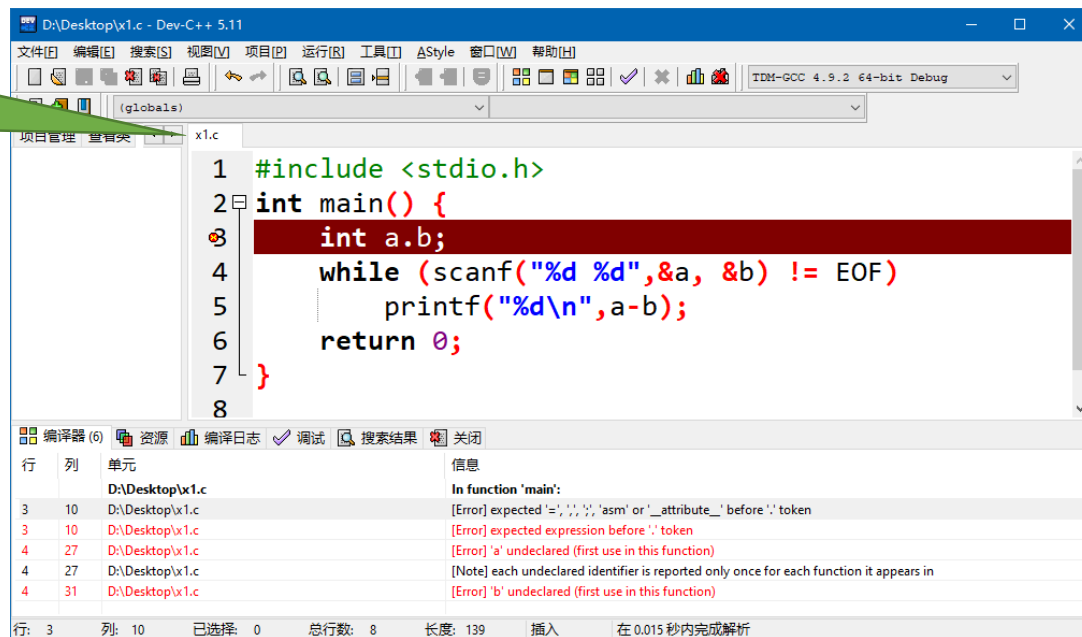


作答步骤

- 编译程序

- 一般用户使用DevC++编译程序（看录像）
- 遇到编译错误应修改至没有错误再提交

用C语言作答时，扩展名勿选用cpp。



The screenshot shows the Dev-C++ 5.11 IDE with a C program named x1.c. The code is as follows:

```
1 #include <stdio.h>
2 int main() {
3     int a.b;
4     while (scanf("%d %d",&a, &b) != EOF)
5         printf("%d\n",a-b);
6     return 0;
7 }
8
```

The compiler output window at the bottom shows the following errors:

行	列	单元	信息
3	10	D:\Desktop\x1.c	In function 'main':
3	10	D:\Desktop\x1.c	[Error] expected '=', ',', ';', 'asm' or '__attribute__' before ':' token
4	27	D:\Desktop\x1.c	[Error] expected expression before ':' token
4	27	D:\Desktop\x1.c	[Error] 'a' undeclared (first use in this function)
4	27	D:\Desktop\x1.c	[Note] each undeclared identifier is reported only once for each function it appears in
4	31	D:\Desktop\x1.c	[Error] 'b' undeclared (first use in this function)

The status bar at the bottom indicates: 行: 3 列: 10 已选择: 0 总行数: 8 长度: 139 插入 在 0.015 秒内完成解析



作答步骤

- 用样例输入测试程序

- 下载样例输入或将其中的蓝底色文字复制到文本文件中
- 在程序主体之前加上freopen语句

```
#include <stdio.h>
int main()
{
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
    (程序主体)
}
```

运行前，确保同级目录下有in.txt文件；运行后，同级目录下将生成in.txt文件。

提交到OJ前，勿忘删除freopen语句

- 运行程序后，将输出文件打开逐字符对比样例输出



作答步骤

- 根据输入范围设计测试数据

- 例如“寻找第二大的数”题，题意大致为寻找给定四个数（1000以内）中第二大数（四个数相同则输出该数）。
- 该题应测试 1000^4 种情况，难以完成。精确地划分测试类又需要花费较多的思考时间，权衡后，建议测试0000~3333共 $4*4*4*4=256$ 种情形。

```
#include <stdio.h>
int main() {
    freopen("in.txt", "w", stdout);
    for (int i = 0; i < 4; i++)
        for (int i2 = 0; i2 < 4; i2++)
            for (int i3 = 0; i3 < 4; i3++)
                for (int i4 = 0; i4 < 4; i4++)
                    printf("%d %d %d %d\n", i, i2, i3, i4);
}
```



作答步骤

- 根据设计好的输入数据测试程序
 - 步骤同“用样例输入测试程序”
 - 如果有条件，做题人应设计对应的“样例输出”以供对照
 - 应保证对应的设计输出是正确的，但计算的过程可以是很慢的
 - 如果没有条件，做题人应手工检验输出是否正确
 - 抽查下界、上界，并在中间抽查出部分数据
 - 手工计算或初步验算结果的准确性
 - 对于发现错误，确认是程序设计错误引起的，应修改程序



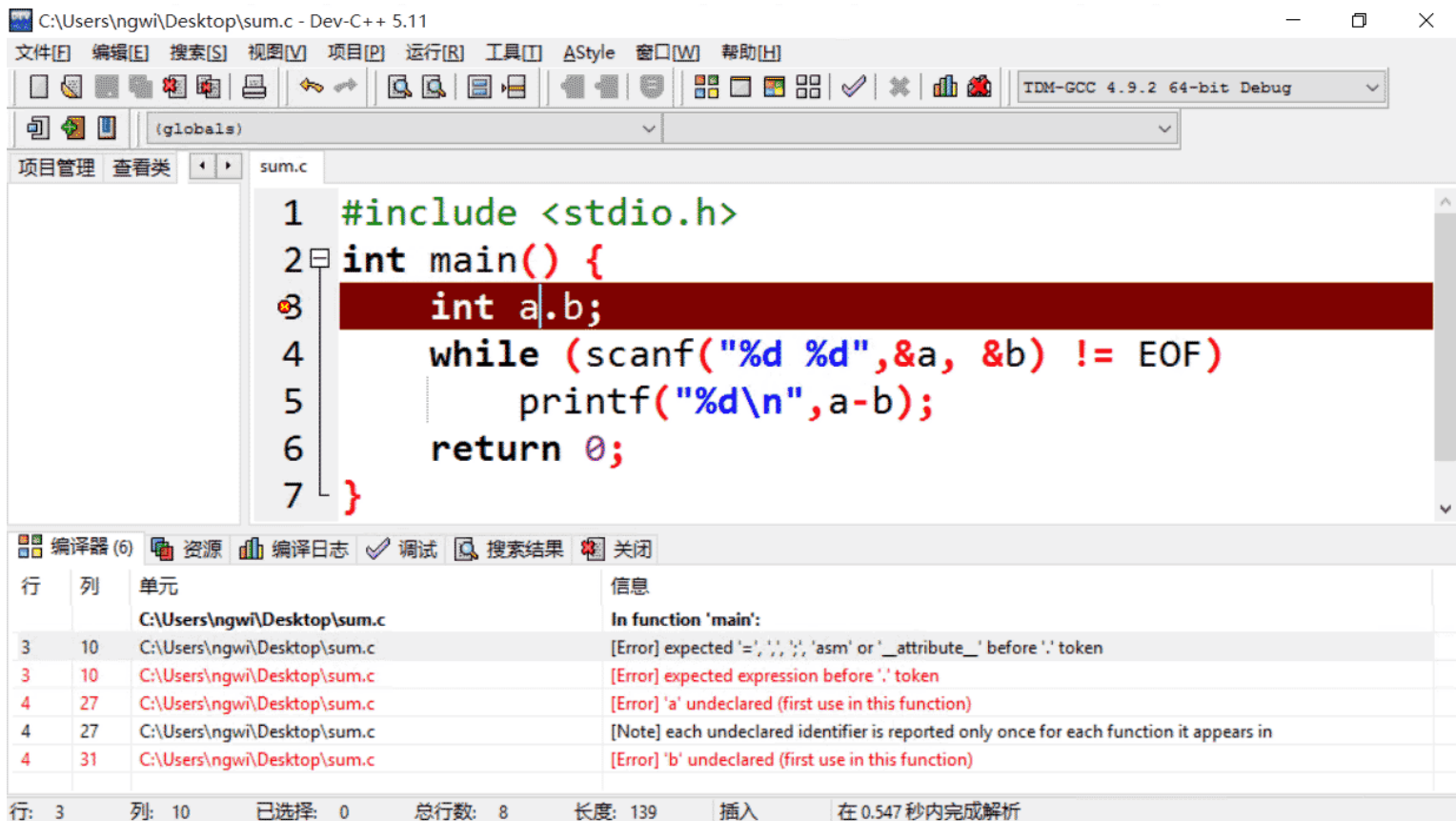
作答步骤

- 在确认不发生以下错误后，提交程序
 - 编译错误、运行错误、答案错误、格式错误
 - 时间超限、内存超限、输出超限
- 否则继续修改优化程序，直到正确



避免“编译错误”

- 当你没有通过编译之前，请勿提交，提交了也是错的。



The screenshot shows the Dev-C++ 5.11 IDE with a C program named `sum.c`. The code is as follows:

```
1 #include <stdio.h>
2 int main() {
3     int a.b;
4     while (scanf("%d %d",&a, &b) != EOF)
5         printf("%d\n",a-b);
6     return 0;
7 }
```

The compiler output window at the bottom shows the following errors:

行	列	单元	信息
		C:\Users\ngwi\Desktop\sum.c	In function 'main':
3	10	C:\Users\ngwi\Desktop\sum.c	[Error] expected '=', ',', ';', 'asm' or '__attribute__' before '.' token
3	10	C:\Users\ngwi\Desktop\sum.c	[Error] expected expression before '.' token
4	27	C:\Users\ngwi\Desktop\sum.c	[Error] 'a' undeclared (first use in this function)
4	27	C:\Users\ngwi\Desktop\sum.c	[Note] each undeclared identifier is reported only once for each function it appears in
4	31	C:\Users\ngwi\Desktop\sum.c	[Error] 'b' undeclared (first use in this function)

At the bottom of the IDE, the status bar shows: 行: 3 列: 10 已选择: 0 总行数: 8 长度: 139 插入 在 0.547 秒内完成解析



避免“编译错误”

• 果不其然，出错了

编译错误的提示信息，如：
Main.c文件第3行第10列出现一个错误

编译错误的提示信息，如：
Main.c文件第4行第25列a没有初始化

信息

```
Main.c: In function 'main':
Main.c:3:10: error: expected '=', ',', ';', 'asm' or '__attribute__' before '.' token
    int a.b;
        ^
Main.c:3:10: error: expected expression before '.' token
Main.c:4:27: error: 'a' undeclared (first use in this function)
    while (scanf("%d %d",&a, &b) != EOF)
                          ^
Main.c:4:27: note: each undeclared identifier is reported only once for each function it appears in
Main.c:4:31: error: 'b' undeclared (first use in this function)
    while (scanf("%d %d",&a, &b) != EOF)
                              ^
```

对上述错误的一些提示

系统提示

错误信息	error: 'a' undeclared (first use in this function)
系统提示	低级语法错误，变量或函数没有定义，注意严谨。



避免“编译错误”

- 告警也很重要

- 告警是编译器对用户可能出现错误的提示
- 由于用户可能故意需要这么做，直接提示编译错误不合适

```
C:\Users\ngwi\Desktop\test-overflow.c - Dev-C++ 5.11
文件[F] 编辑[E] 搜索[S] 视图[V] 项目[P] 运行[R] 工具[T] AStyle 窗口[W] 帮助[H]
(globals)
项目管理 查看 test-overflow.c
1 #include <stdio.h>
2 int main() {
3 char a=251, b;
  b=a*3;
  printf("%d\\n", a-b);
  return 0;

C:\***\test-overflow.c: In function 'main':
C:\***\test-overflow.c:3:12: warning: overflow in
implicit constant conversion [-Woverflow]
char a=257, b;
      ^

此处说明用户赋值时溢出了

行: 3 列: 15 已选择: 0 总行数: 8 插入 在 0.032 秒内完成解析
```

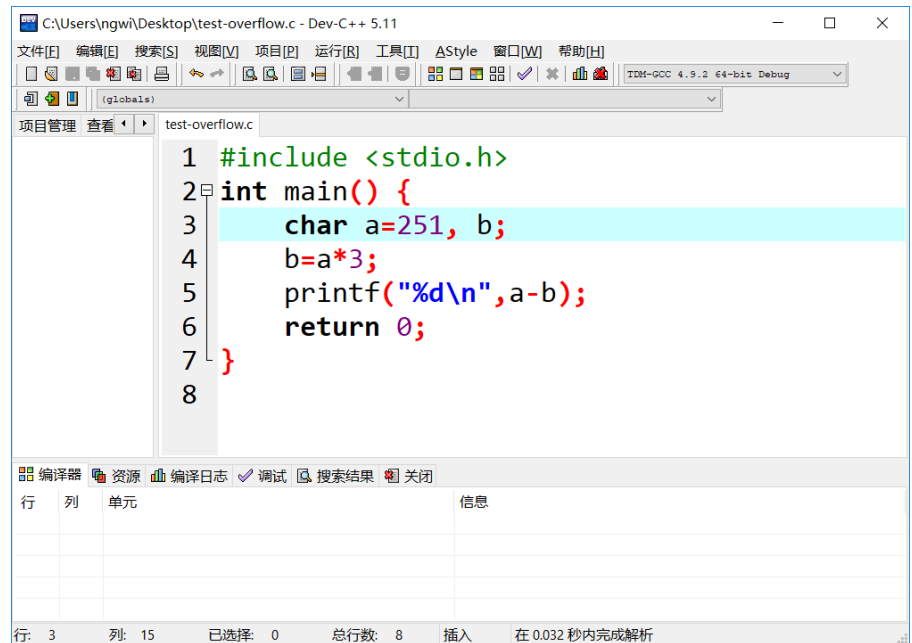


避免“编译错误”

- 编译成功

- 程序没有输出错误信息

- 程序没有输出告警信息或告警信息不重要



```
1 #include <stdio.h>
2 int main() {
3     char a=251, b;
4     b=a*3;
5     printf("%d\n",a-b);
6     return 0;
7 }
8
```



避免基本的“答案错误”

- 提交之前应先用样例输入测试程序

- 样例输入和输出只是帮助理解题意

- 答案是3，而你的输出是1，强行提交还是答案错误。

- 后台数据的坑是程序测试的考点，考谁的测试更全面。

```
#include <stdio.h>
int main() {
    int a,b;
    while (scanf("%d %d",&a, &b) != EOF)
        printf("%d\n",a-b);
    return 0;
}
```

```
Test Input (top 50 lines):
1 2
Difference (top 100 lines):
@@ -1 +1 @@
-3
\ No newline at end of file
+-1
```



答案差异文件的示意

- 单击“状态”页的“答案错误”标记或下载报告获得

```
Test Input (top 50 lines):  
1 2  
Difference (top 100 lines):  
@@ -1 +1 @@  
-3  
\ No newline at end of file  
+-1  
Test Input (top 50 lines):  
3 3  
Difference (top 100 lines):  
@@ -1 +1 @@  
-6  
\ No newline at end of file  
+0
```

测试输入的前50行

参考输出的前100行

输入文件的结束标记

此处分割了两个测试组

```
Test Input (top 50 lines):  
1 3  
Difference (top 100 lines):  
@@ -1 +1 @@  
-3  
\ No newline at end of file  
+-1  
Test Input (top 50 lines):  
1 3  
Difference (top 100 lines):  
@@ -1 +1 @@  
-4  
\ No newline at end of file  
+-2
```

提示中参考输出的首行是第1行

提示中用户输出的首行是第1行

从首行开始参考输出是3

从首行开始用户输出是-1



答案差异文件的示意

- 单击“状态”页的“答案错误”标记或下载报告获得

Test Input (top 50 lines):

1 2

Difference (top 100 lines):

@@ -1 +1 @@

-3

\ No newline at end of file

+ -1

行号	参考输入	用户输出
1	1 2	1 2

行号	参考输出	用户输出
1	3	-1

这里是文件结尾，scanf在这里返回EOF

Test Input (top 50 lines):

60.12 3000.49 95.75

Difference (top 100 lines):

@@ -1,2 +1,2 @@

--0.0319

--49.8764

+-115.4184

+-180274.0469

行号	参考输入	用户输出
1	60.12 3000.49 95.75	60.12 3000.49 95.75

行号	参考输出	用户输出
1	-0.0319	-115.4184
2	-49.8764	-180274.0469



避免“运行错误”

- 运行错误一般是没有很好地测试边界情况所导致的。
 - 例如程序要求输入是长度为100的字符串，你测试时却仅限于样例输入，只测试长度为2的字符串，这是远远不够的。
- 还有可能你调用了系统函数，可能损坏OJ系统。
 - `gets()`、`puts()`、`getch()`、`putchar()`、`system()`
- OJ不说明发送运行错误的具体实例。

Runtime Error: Floating point exception
Runtime Error: Segmentation fault

浮点错误，检查是否有除以零的情况。

段错误，检查是否有数组越界，指针异常，访问到不应该访问的内存区域。



避免“时间超限”

- 明显的情况是：程序在测试输入情况下死循环了。
 - 一个情况是你的程序会死循环
 - 另一个情况是，你使用了错误的输入格式。
 - 例如程序要求输入是“3 5”，而你自行测试的时候误以为输入“3,5”，这种情况下，输入“3 5”反而会产生错误。
 - 例如程序的最后一行是以“EOF”结束的，而你自行测试的时候是以回车结束，并没有Ctrl+Z模拟“EOF”，也会产生错误。



避免“时间超限”

- 排除上述情况，很可能你的算法不够好，导致在大数据的测试下程序明显超时了。
 - 例如：求 $n!$ 的最后一位，显然在 $5!$ 及以后答案就是0了。可是你的程序一个个算，最后虽然也输出0，但在测试数据 $n=2000000000$ 的时候，很可能时间超限了。



避免“时间超限”

- 如果你的算法不够好，你可以找一个正确的代码，通过测试大量数据（或者少量循环多次），直观感受耗时的差异。
 - 可以在一些循环里加上计数器`count++`，然后打印一下`count`，两个程序都打印，做出对比。
 - 只有这样，你才能服膺，才能静下心来看看别人的代码和自己的代码有什么区别。



避免“内存超限”

- 有时候你动态开辟空间却忘了释放它，经过几万次的循环以后内存就超限了。
- 排除上述情况后，很可能你对空间的把控不够好，导致在大数据的测试下程序明显超限了。
 - 例如：求巨型整数的乘法，你图方便开辟了一个 $1024*1024$ 的整型数组，已接近4MB。当只存储一位十进制数字时，存储效率是0.1。
- 这是考点，考谁的解题方法更好。



避免“格式错误”

- 认真审题，判断程序有多少个空格输出

1 6 5

这里多出个空格

1 6 5

这里是1、空格、6、空格、5

样例输入

1 -6 5

样例输出

1.0000 5.0000

- 高级教程

– 将你的输出重定向到文件里（选中你的输出）看看有没有空格或回车，来进行比较。



避免“格式错误”

- 尽量不要再使用键盘输入的方式测试数据
 - 因为你有可能按了回车键而没注意到这个回车是需要用 `printf` 输出的，不是手动添加的
 - 重定向更接近实际。
- 问题来了，如何重定向呢？
 - 你需要把你的测试数据存到一个文件里。



终极避免“答案错误”

- 解决避免“答案错误”的问题需要用到“对拍法”。
 - 由于测试数据未知，先编写一个代码简单但可能无法计算大数据或者耗时特别长的（但一定是答案正确）的程序
 - 然后用随机数生成题目规定的输入，用这个一定正确的程序输出得到参考答案。
 - 然后你正式做题的时候，讲你的程序与这个输入输出做比较，直到找到你程序的大部分缺陷。



终极避免“答案错误”

- OJ提供了答案错误查询功能，但不要依赖于答案。
 - 不要觉得答案错了我就改吧，改到对了就对了。
 - 不要觉得AC是第一位的，独立完成独立思考才是第一位的。
 - 答案错误开放查询是让你发现你在思考问题的时候会在哪些地方思虑不周，以期改善，不是让你以答案为终点。
- 只会写一道题是没有用的。



新手可能有的错误

- 不经测试直接提交代码
- 重复提交代码：重复的代码判定结果是一样的
 - 怀疑OJ出错时
 - OJ出现长时间等待时



示例



解题示例：阶乘末六位

- 题目描述

- 求 $x!$ 的末六位。

- 输入

- 一个整数 x ， $0 < x < 2^{31} - 1$ 。

- 输出

- $x!$ 的末六位，不足以0在数字左侧填充。

样例输入	样例输出
1	000001
2	000002
3	000006



解题示例：阶乘末六位

- 第一步：拿过一张草稿纸。在草稿纸上比划。

– 阶乘是主要公式

$$n! = \prod_{i=1}^n i$$

```
for (i = 1; i <= n; i++)  
    res = res*i;
```

– 支持多行输入

```
while (scanf("%d", &n)==1)  
{  
    ...  
}
```

– 格式输出

```
printf("%06d", res);
```

- 不要去管细节，否则就没完没了了。



解题示例：阶乘末六位

- 将上述部分搭成一个初步的程序

```
#include <stdio.h>
```

```
int main(void)
```

```
{
```

```
while (scanf("%d", &n)==1)
```

```
{
```

```
for (i = 1; i <= n; i++)  
    res = res*i;
```

```
printf("%06d", res);
```

```
}
```

```
return 0;
```

```
}
```

变量n在这里声明

声明语句应放在最小的作用范围内。
变量i和res应在这里声明

res在使用前应初始化。此处
是阶乘，因而初始化应为1。

从题意看，输出末六位

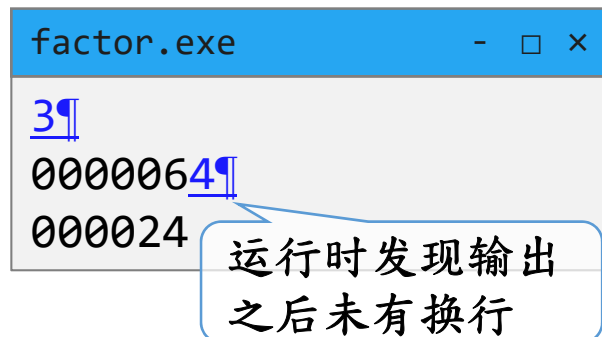


解题示例：阶乘末六位

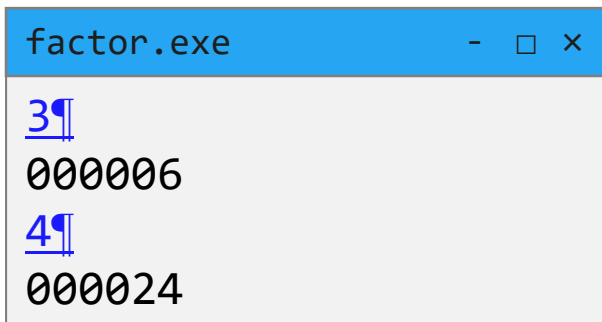
- 将上述部分搭成一个初步的程序

```
#include <stdio.h>
int main(void)
{
    int n;
    while (scanf("%d", &n)==1)
    {
        int i, res = 1;
        for (i = 1; i <= n; i++)
            res = res*i;

        printf("%06d", res % 1000000);
    }
    return 0;
}
```



运行时发现输出之后未有换行



解题示例：阶乘末六位

• 考虑如何测试数据

```
#include <stdio.h>
int main(void)
{
    int n;
    //while (scanf("%d", &n))
    for (n = 1; n < 20; n++)
    {
        int i, res = 1;
        for (i = 1; i <= n; i++)
            res = res*i;

        printf("%2d: %06d\n", n, res % 1000000);
    }
    return 0;
}
```

这里稍作修改，用于测试程序

这里稍作修改，用于测试程序

factor.exe

1:	000001
2:	000002
3:	000006
4:	000024
5:	000120
6:	000720
7:	005040
8:	040320
9:	362880
10:	628800
11:	916800
12:	001600
13:	053504
14:	945280
15:	310016
16:	189184
17:	-522240
18:	-433024
19:	641728

连负数都出来了
整数溢出了。



解题示例：阶乘末六位

• 测试更多的数据

```
#include <stdio.h>
int main(void)
{
    int n;
    //while (scanf("%d", &n))
    for (n = 1; n < 100; n++)
    {
        int i, res = 1;
        for (i = 1; i <= n; i++)
            res = res*i % 1000000;

        printf("%2d: %06d\n", n, res);
    }
    return 0;
}
```

再次修改，测试更多的数据

```
factor.exe  - □ ×
20: 640000
21: 440000
22: 680000
23: 640000
24: 360000
25: 000000
26: 000000
27: 000000
28: 000000
29: 000000
30: 000000
31: 000000
```



解题示例：阶乘末六位

- 把输入输出换回，提交到OJ

— 发现超时了。（应该是输入太大）

```
#include <stdio.h>
int main(void)
{
    int n;
    while (scanf("%d", &n))
    {
        int i, res = 1;
        for (i = 1; i <= n; i++)
            res = res*i % 1000000;
        printf("%06d\n", res);
    }
    return 0;
}
```

可能是n太大算太久。在n大于24时，直接输出0更快。

```
factor.exe  - □ ×
20: 640000
21: 440000
22: 680000
23: 640000
24: 360000
25: 000000
26: 000000
27: 000000
28: 000000
29: 000000
30: 000000
31: 000000
```



解题示例：阶乘末六位

- 改正后，提交到OJ，获得AC

```
#include <stdio.h>
int main(void)
{
    int n;
    while (scanf("%d", &n))
    {
        int i, res = 1;
        if (n > 24)
            res = 0;
        else
            for (i = 1; i <= n; i++)
                res = res*i % 1000000;
        printf("%06d\n", res);
    }
    return 0;
}
```



解题示例：大整数加法

- 题目描述

- 求不超过31位大整数之和。

- 输入

- 两个整数 a, b ， $0 < x < 2^{31} - 1$ 。

- 输出

- $c = a + b$ 。

样例输入	样例输出
1 2	3

这样的例子，不要也罢



解题示例：大整数加法

- 自己先解一个大整数加法感受一下，总结规律。

$$\begin{array}{r} a \quad 05798676521346745697 \\ b + 04321021897988765435 \\ \hline c \quad 10119698419335511132 \end{array}$$

Diagram illustrating the addition of two large integers a and b to produce c . The numbers are aligned by their least significant digits (right-aligned). Red dashed boxes highlight the alignment of the digits. Callouts indicate the indices of the digits being added: $a[a_{len}-1]$ and $b[b_{len}-1]$ for the least significant digits, and $c[0]$ for the first digit of the result.

- 规律

— 从最后一位开始算起

先不要纠结具体定位，
先定框架再微调

```
alen = strlen(a);
blen = strlen(b);
for (i = 1; i < alen; i++)
{
    a[alen - i] += b[alen - i];
}
```



解题示例：大整数加法

- 加上框架与测试框架
 - 基本加法
 - 输出
 - 基本的测试集



```
#include <stdio.h>
#include <string.h>
#define SIZE 100
int main(void)
{
    const char a[SIZE] = "57";
    const char b[SIZE] = "43";
    const char cr[SIZE] = "100";
    char c[SIZE] = { 0 };
    size_t alen = strlen(a), blen = strlen(b), clen = alen + 1, i;
    for (i = 1; i <= alen; i++)
        c[clen - i] = a[alen - i] - '0' + b[blen - i] - '0';
    for (i = 1; i < clen; i++)
        printf("%2d,", c[i]);
    printf("\n");
    for (i = 0; i < clen; i++)
        printf("%2c,", cr[i]);
    printf("\n");
    return 0;
}
```



解题示例：大整数加法

- 改进进位的错误

程序输出

0, 9, 10

进位错误

参考输出

1, 0, 0

```
for (i = 1; i < alen; i++)  
{  
    c[clen - i] += a[alen - i] - '0' + b[blen - i] - '0';  
    if (c[clen - i] >= 10)  
    {  
        c[clen - i - 1]++;  
        c[clen - i] = c[clen - i] - 10;  
    }  
}
```

此处应为+=，使进位生效

进位处理



解题示例：大整数加法

• 测试更多情况

a	b	c	程序输出c	可能的问题
557	43	600	0, -42, 0,	没有考虑a比b长

```
#define MAX(A,B) ((A)>(B)?(A):(B))
size_t maxlen = MAX(alen, blen);
size_t clen = maxlen + 1, i;
for (i = 1; i <= maxlen; i++) {
    if (alen >= i && blen >= i)
        c[clen - i] += a[alen - i] - '0' + b[blen - i] - '0';
    else if (alen >= i)
        c[clen - i] += a[alen - i] - '0';
    else
        c[clen - i] += b[blen - i] - '0';
    if (c[clen - i] >= 10) {
        c[clen - i - 1]++;
        c[clen - i] = c[clen - i] - 10;
    }
}
```

求A,B最大值的宏，也可以用if实现

clen应该是maxlen+1（第一位是0可以不打印）

由于alen是无符号，因此不能判断alen-i>=0，它永远成立。



解题示例：大整数加法

- 首位为0不输出

```
for (i = 0; i < clen; i++)  
{  
    if (i != 0 || c[i] != 0)  
        printf("%d", c[i]);  
}
```

- 多测试几种情况

a	b	c	程序输出c	可能的问题
557	43	600	600	a比b长
43	523457	523500	523500	b比a长
193	457	650	650	ab等长
590	653	1243	1243	需要进位
102	4353	4455	4455	不需要进位



解题示例：大整数加法

- 可以更多的测试
 - 两层循环生成不同的a和b
 - 答案生成
- 整理代码准备提交



```
#include <stdio.h>
#include <string.h>
#define SIZE 100
#define MAX(A,B) ((A)>(B)?(A):(B))
int add(const char a[], const char b[], char c[])
{
    size_t alen = strlen(a), blen = strlen(b);
    size_t maxlen = MAX(alen, blen);
    size_t clen = maxlen + 1, i;
    for (i = 1; i <= maxlen; i++)
    {
        if (alen >= i && blen >= i)
            c[clen - i] += a[alen - i] - '0' + b[blen - i] - '0';
        else if (alen >= i)
            c[clen - i] += a[alen - i] - '0';
        else
            c[clen - i] += b[blen - i] - '0';
        if (c[clen - i] >= 10)
        {
            c[clen - i - 1]++;
        }
    }
}
```



```

        c[clen - i] = c[clen - i] - 10;
    }
}
return clen;
}

int main(void)
{
    long long int a, b;
    char sa[SIZE], sb[SIZE], sr[SIZE];
    size_t i;
    for (a = 0; a < 10; a++)
    {
        for (b = 0; b < 10; b++)
        {
            char sc[SIZE] = { 0 };
            int offset = 0;
            sprintf(sa, "%lld", a);
            sprintf(sb, "%lld", b);
            sprintf(sr, "%lld", a + b);

```



```

    int clen = add(sa, sb, sc);
    for (int ci = 0; ci < clen; ci++)
        sc[ci] += '0';
    sc[clen] = 0;
    if (sc[0] == '0')
        offset = 1;
    printf("%s + %s = %s (Yours: %s)", sa, sb, sr, sc +
offset);

    if (strcmp(sc + offset, sr) == 0)
        printf("OK");
    else
        printf("ERR");
    printf("\n");
}
}
return 0;
}

```



解题示例：大整数加法

- 最终整理代码准备提交



```
#include <stdio.h>
#include <string.h>
#define SIZE 100
#define MAX(A,B) ((A)>(B)?(A):(B))
int main(void)
{
    char a[SIZE], b[SIZE];
    while (scanf("%s %s", a, b) == 2)
    {
        char c[SIZE] = { 0 };
        size_t alen = strlen(a), blen = strlen(b);
        size_t maxlen = MAX(alen, blen);
        size_t clen = maxlen + 1, i;
        for (i = 1; i <= maxlen; i++)
        {
            if (alen >= i && blen >= i)
                c[clen - i] += a[alen - i] - '0' + b[blen - i] - '0';
        }
    }
}
```



```

    else if (alen >= i)
        c[clen - i] += a[alen - i] - '0';
    else
        c[clen - i] += b[blen - i] - '0';
    if (c[clen - i] >= 10)
    {
        c[clen - i - 1]++;
        c[clen - i] = c[clen - i] - 10;
    }
}
for (i = 0; i < clen; i++)
{
    if (i != 0 || c[i] != 0)
        printf("%d", c[i]);
}
printf("\n");
}
return 0;
}

```



解题示例：大整数加法

- 事情远没有结束
- 题目要求：不超过20位十进制数
 - long long int无能为力
 - 但long double可以表示
 - 别忘了还可以用长浮点型表示整数



```
#include <stdio.h>

int main()
{
    long double x, y;

    while (scanf("%11f%11f", &x, &y) != EOF)
    {
        printf("%.011f\n", x + y);
    }

    return 0;
}
```



C语言程序设计

E2



谢谢

厦门大学信息学院软件工程系

黄炜 副教授