# 通关考攻略

## 一．什么是通关考

### 1.产生背景

敲代码手不稳是个大毛病，往往会让一份能AC的代码变成99.995%正确，失之毫厘谬以千里，最近十场个人赛很少有能一次AC的经历，仔细想想除了根本逻辑上的错误，大概都是跪在这些细节上：

1.输出格式，输入格式是否符合规范，有没有Case #？是否有多余空格输出？I64d or lld？输出浮点数尽量不要用cout。

2.i和j，n和m，l和r有没有写混了的。。（今晚检查了两小时的程序发现i<=m写成了i<=n...吾之内心几乎崩溃orz。。

3.边界问题，从0开始还是从1开始？递推式有没有越界？不要使用cnt数组来累计某个值出现多少次，最好使用map来映射。。

4.爆int，爆longlong，数论题尤其要注意，每次测试滚键盘来几个大数字。爆int的话实在拿不准数据范围，在内存允许的情况下#define int long long；爆long long的话把每个乘法操作模除大素数（必要情况下加法操作最好也要。

5.对于不合法输入或无解情况的处理。。是输出-1还是0还是no solution好好读题。

6.某些应该注释掉的句子没注释，freopen什么的。。

7.特判情况遗漏。

8.数据还没完全读入就贸然结束程序。。虽然有些时候答案已经出来了。

解决办法还是静下心来，思路乱的时候手不要碰键盘，太庞大太复杂的流程分模块来写，先想好样例再去写代码。

眼下组队赛要开始了，不能坑队友。

debug流程（from知乎：

1、重新通读一遍代码，检查初始化，检查输出格式，需不需要输出CASE等

2、检查经常出现的一些手误，两重for循环中i,j用混，多重for循环中i被多次使用，long long相关问题等。

3、检查数据范围，重新读一遍题，确认题意，思考边界数据对代码的影响，数组是否开小，是否爆int等。

4、进行暴力对跑，写个暴力但保证正确的版本，利用小数据和代码对跑。

### 3.使用平台

通关考是在程序设计竞赛提交平台上进行。若要通关，必须熟悉提交平台。在平台上练习做题，才有可能取得好的考试成绩。

浙江工业大学的通关考试提交平台有两个：编程教学实验平台cpp.zjut.edu.cn；程序设计竞赛平台acm.zjut.edu.cn。两者功能相似，题库发展的侧重点有所不同，一个面向教学和教育部优秀教材的全国读者，一个面向浙工大编程训练与竞赛。通关考每场学生人数超200人，为缓解网络负载，视实际情况决定是否允许两个提交系统同时工作。

通关考的题库范围标识（见《通关练习题集》），以cpp.zjut.edu.cn为准。学生平时可以在任何一个平台上练习做题。网络通关模拟考试同时在两个平台中开放。

//=====================================

// comp.cpp

// 组合数

//=====================================

#include<iostream>

using namespace std;

//-------------------------------------

int a[34][34];

//-------------------------------------

int main(){

for(int i=0; i<34; i++){

a[i][0]=1; // 每行第1个元素初始化

for(int j=1; j<=i; j++)

a[i][j] = a[i-1][j-1]+a[i-1][j];

}

for(int n,m; cin>>n>>m; )

cout<<a[n][m]<<"\n";

}//------------------------------------

## 二．报名考试和成绩

### 1.报名

通关考所需知识内容，在第一学期《程序设计基础C》已全覆盖。所以，在第二学期举办的通关考试，完全是考核每个学生学以致用的能力。通关考第一场，每个学生必须参加，而对于第二场通关考，可以视自己的学习情况选择报名或放弃。第一场通关考结束后一周，可以在acm.zjut.edu.cn网上报名参加第二次通关考。

通关考实行两次通关，成绩取其高规则，鼓励未通关者、考得不理想者再次尝试，参加第二次通关考。

### 2.考试

#### 考试形式

通关考形式为以计算机机房作为考场，以程序设计竞赛网站作为提交操作平台的开卷上机考试。

#### 公平性

基于计算中心考场现状，通关考安排为，一个考生一张桌子两个电脑。

机房作为考试场地，每个机房将有至少两个监考教师。另外设有一个总监考，负责全场考试问题的最终解释。

印制和分发试卷，主要是为了减少网路访问流量，考试期间，有系统维护员来保证考试网络提交顺畅。同时，书面形式的试卷也减缓了电脑视觉伤害，并方便考后题目讨论和查验。

考试过程中每个学生都有唯一的一个用户名和密码，用来表示学生的身份和对应其成绩。

#### 作弊防范与处理

通关考作为C++程序设计课程设计考核的一部分，应严格参照浙江工业大学课程考核要求进行，上机考试过程中如有发现疑似作弊行为，一经认定，将严格参照《浙江工业大学学生课程考核规则与违纪处分办法》进行处理。

任何泄漏自己用户名和密码的行为被视为作弊。用户名和密码传递进行作弊的企图，将在内部提交系统的提交机器MAC编号上取证。

考试全程开启视频监控，并按照学校要求存档。

#### 公正性

通关考成绩是通过所做的题数反映的，每个学生的通关成绩可以实时的在提交系统中公开看到。

### 3.成绩

考试成绩分别为50分（满分）、45分、35分、0分。分别对应做出5题，4题，3题，2题及以下。

参加考试的学生必须在考试中，在5个考题中做出3题或以上（多多益善）才达到通关要求（35分是通关底线）。做出2题或以下即没有通关，需要尝试下一次通关考。

通关成绩越高，则其课程成绩必然越好。如果下一次仍未通关，则需要在重修《C++程序设计课程设计》课程，重新得到通关考机会。具体参考《通关考方案》。

## 三．把握通关难度

### 1.难度系数

通关考题目的难度系数分A~C级。

|  |  |
| --- | --- |
| 难度 | 描述 |
| A | 纯粹无输入的HelloWorld型，含多行输出情形，含特殊字符输出情形 |
| B | 读入单一数据，分情况简单表达式输出；循环读入简单数据，循环统计或汇总计算，涉及简单字串，简单输出三角形、矩形等字符形状 |
| C | 多重循环，需用数组，涉及字串简单搜索，数组排序等简单计算，需要认真考量输入数据结构，或者需要简单定义函数 |

考题难度是编程能力的衡量指标，涉及编程问题的分析、设计、实现的难度，其针对原创题而言。对于见到过、做到过的题库题，难度只作参考依据。

### 2.难度影响因素

#### 陷阱题

陷阱是指题目描述中没有明确说明，通常人们会这样理解，而实际上却不是的一种误导性描述。例如cpp.zjut.edu.cn中1009题（简单编码）希望对字串中的字符进行改写。该题目没有输入和输出描述，要不是后来加了提示信息，还真以为每个字串中没有空格的呢。字串含空格与不含空格，代码设计会相差很大，所以在给出提示信息之前，通过此题的人很少。

陷阱题带给练习者或考试者许多无谓的时间浪费，它对培育编程兴趣的初学者来说是一种负面干扰。所以对于通关考试来说，严格杜绝陷阱题。

对于提交平台中的题库，存在陷阱题却是自然的。编程训练不光是针对通关，而是对于各种学习编程的人群。许多人已经具备浓厚的编程兴趣，不为陷阱的困惑所动，反而从中提炼出斗志。

训练理解能力并非基于精确化和理想化的题目描述，而是需要从题目描述的各种因素中综合地把握题目的要求。对于竞赛，也确实需要很多理解能力的训练和经验的积累。

#### 英文题

通关考暂不含英文题。

### 3.不确定性说明

#### 难度制定问题

难度系数是题目设计的预计难度与根据考试或竞赛中的通过率统计出来的实际难度相结合而评定的。

#### 提交系统现状

浙工大的提交系统用于校级以下各级竞赛和程序设计课程考试。其题库题的难度既需要满足校级程序设计竞赛和亚洲区域赛训练要求，又需要满足课程教学的要求，所以题库中各种难度的题目并存。

浙工大提交平台中除了因赛事而扩充题库外，在长期从事的上机实验和考试实践中，积累了大量适合考试的题库题。所以，与别的学校的提交平台不同，浙工大的提交平台中，低难度的题目比较多，中文题目比较多，更适合于课程教学的练习和考试。

#### 考试难度区间

首先要知道，题库题中的题目难度系数是为考试目的而标示出来的。

通关考是面向计算机大类程序设计课程的学生。作为课程考试的一部分，其难度设计的区间在A类与C类之间。

由于难度系数在制定中存在不确定性和难度影响因素，所以，学生在练习和考试中可能会觉得实际难度与对难度理解和判断的感觉不符。

## 四．做题旨要

### 1.考题结构

题库题1题（A难度）

原创题3题（B难度）

原创题1题（C难度）

### 2.B类题例解

#### 题目例子

区域平方和

基本描述

给定一个整数区域，计算其区域内全部整数的平方和。

输入描述

一些正整数对A，B（1<A≤B<1000）。表示正整数区域[A,B]。

输出描述

对每个正整数对A，B，以一行的形式输出其区域所有整数的平方和。

样本输入

1 3

2 5

样本输出

14

54

#### 分析

##### 数据区域

题目中的输入描述，给出了一些数据区域：整型数的闭区间[A,B]，其中1<A≤B<1000，计算是以数据区域为单位，闭区间中所有的整数都是挨个儿连续的，所以，对于给定的整型数A和B，其数据取值为A , A+1, A+2, … , B-1, B。

##### 循环获取数据

因为有很多整数对A和B，为了获取这些整数对，需要对两个整数变量进行循环读取：

**for**(**int** A,B; cin>>A>>B; )

{

// 处理该A，B区间的计算

}

##### 循环累计计算

对于已知的A、B，处理该区间的计算，要遍历A、B中所有的整数，故只要循环：

**for**(**int** i=A; i<=B; i++)

{

// 处理某个整数

}

处理某个整数，即是累计该整数的平方。

为了求[A,B]区间的平方和，需要在循环遍历之前，先预设一个累计变量sum，并初始化为0。即：

**int** sum = 0; // 设置求和变量=0

**for**(**int** i=A; i<=B; i++) // 从区间A到B循环每个整数i

{

sum += i\*i; // 累计平方数

}

cout<<sum<<"\n"; // 循环结束后输出汇总数

#### 代码设计

##### 版本一

对于给定的闭区间[A,B]，能够知道有多少个整数，只要对这些连续的整数，循环求平方和即可得到结果。将上面的分析汇总，得到代码为：

//=====================================

// 区间平方和 SumOfRangeSquares1.cpp

// 难度B

//=====================================

*#include<iostream>*

**using namespace** std;

//-------------------------------------

**int** main(){

**for**(**int** A,B; cin>>A>>B; ) // 循环读入A,B区间

{

**int** sum=0; // 设置求和变量=0

**for**(**int** i=A; i<=B; i++) // 从区间A到B循环每个整数

sum += i\*i; // 累计平方数

cout<<sum<<"\n"; // 循环结束后输出汇总数

}

}//====================================

##### 关于代码的注释

标准代码解答往往是难度定级的依据，所以在解答代码的开始位置，除了标注以题目命名的代码文件名，还应该注释难度等级。

但是对参加通关考试的考生来讲，这些注释完全没有必要。

##### 版本二

对于这个简单题，应该能想到求连续数的平方和的计算公式：

12+22+32+…+n2 = n(n+1)(2n+1) / 6

于是从a到b的累计平方和计算为：

a2+(a+1)2+(a+2)2+…+(b-1)2+b2 = [ b(b+1)(2b+1) - (a-1)a(2a-1) ] / 6

由于用一个公式直接代替了循环计算，所以代码设计便可不需要循环，直接利用公式而得到计算结果：

//=====================================

// 区间平方和 SumOfRangeSquares2.cpp

// 难度B

//=====================================

*#include<iostream>*

**using namespace** std;

//-------------------------------------

**int** main(){

**for**(**int** a,b; cin>>a>>b; )

{

cout<<(b\*(b+1)\*(2\*b+1)-(a-1)\*a\*(2\*a-1))/6<<"\n";

}

}//====================================

#### 难度定位

任何问题都会涉及循环读入数据，有的是单个整数或浮点，有的是成组的整数（像这里的A、B区间），等等。

获取了数据之后，就要对数据进行计算。所以是在循环读入数据的循环体中进行的处理。

所涉及的计算，属于这样的简单循环求和，就定格为B类难度题了。

### 3.C类题例解

#### 题目例子

哪张是大王

基本描述

放在一起的54张扑克牌，按顺序把奇数位上的牌拿走，即拿走第1,3,5,7,9...等牌张。在剩下的牌中再把奇数位的牌拿走，...，如此下去，直到最后只剩下一张牌，这张牌就是“大王”。那么这张牌在原来54张牌中处于第几张的位置呢？经过演算，发现是第32张。

牌张扩大到任意正整数。现在有一叠牌，数清楚牌张，总能通过奇数位牌张拿走的有限轮操作，到达最后只剩下一张牌的状态。假如剩下的这张牌就是大王，问该大王在整个一叠牌中的位置。

输入描述

若干个正整数N（0<N<10000000），每个正整数表示一叠牌的张数。若为0，则结束。

输出描述

对于每个正整数N所表示的牌张，以一行方式输出大王的位置。

样本输入

54

0

样本输出

32

#### 分析

##### 读取数据

该题是对于每次读取的整数N（一叠牌的张数），计算大王的位置。

整数N的取值范围在0<N<10000000，所以，整型int类型足可表示。

因为涉及如果读入一个0的话就结束运行，所以读入数据的循环可以在读入的同时判断数值是否为0：

**for**(**int** a; cin>>a && a; ) // 循环读入牌张数a, 遇到0便结束

{

// 在a张牌中计算大王位置

}

##### 判断大王位置

可以先从54张牌的情况开始分析，因为已知正确答案是32。

再取10张、20张、30张、40张，手工分析大王的位置，分别得到第8、第16、第16、第32的位置。

由此归纳，已知牌张数N，其大王的位置在2max(X)≤N。即不大于牌张数N的2的最大幂次方。例如，20这个牌张数，16是2的幂次的最大数，牌张数若为30，16仍然是2的幂次的最大数。对于54来说，32是2的幂次的最大数。有时候用手工方法从简单数开始分析是有效的。

##### 计算过程设计

根据一个正整数N（牌张数），计算其不大于该数值的最大2的幂次方数，可以设计一个函数来实现：

**int** daWang(**int** n) // 给定牌张数n，返回整型数（大王的位置）

{

**for**(**int** i=1; ; i\*=2) // i初值为1, 以2的次幂逐次递增，无止境

{

**if**(i\*2 > n) // 不断试探i乘2大于n的时刻

**return** i; // 此时i就是大王的位置

}

}

##### 主函数框架设计

因为程序都是从主函数main开始启动运行的，所以在有了函数计算的设计后，就需要在主函数中调用该函数：

**int** main()

{

**for**(**int** a; cin>>a && a; ) // 循环读入牌张数a, 遇到0便结束

cout<<daWang(n)<<”\n”; // 调用函数daWang(a)获得大王位置

}

#### 代码设计

##### 版本一

合并函数设计和主函数框架设计这两段代码，得到完整的代码设计：

//=====================================

// 哪张是大王 WhichIsKing1.cpp

// 难度C

//=====================================

*#include<iostream>*

**using namespace** std;

//-------------------------------------

**int** daWang(**int** n){

**for**(**int** i=1; i<=n; i\*=2)

**if**(i\*2 > n)

**return** i;

}//------------------------------------

**int** main(){

**for**(**int** a; cin>>a && a; )

cout<<daWang(a)<<”\n”;

}//====================================

##### 版本二

“大王在哪里”一题还可以采用位操作的方法，对每个牌张数进行二进制的位搜索，来查找大王位置：

//=====================================

// 哪张是大王 WhichIsKing2.cpp

// 难度C

//=====================================

*#include<iostream>*

**using namespace** std;

//-------------------------------------

**int** main(){

**for**(**int** a; cin>>a && a; )

{

**int** i = 1; // i初值为1，跟踪查找不大于a的最大2次幂

**while**(a&i != i) i<<=1; // a与i做位与操作，即忽略a的低位

cout<<i<<"\n";

}

}//====================================

或许该代码具有更好的运行性能，但从知识点角度说，仍归于C级难度。

#### 难度定位

该题从代码实现的难度上来说，并不比B类题大多少。

但是，因为大王的位置计算，首先需要一番分析，才能找到位置规律，然后找到计算方法，再用代码实现。这个额外的分析过程，被认为是超过了B类题的难度。

又因为所使用的编程知识点（简单循环，计算，使用了函数定义和调用）多了函数的使用。当然可以完全不用函数调用，但那样的话，循环计算的复杂性将增加，诸多因素会影响提交通过率。

毕竟没有涉及其他数组，以及与数组有关的搜索、排序等计算，所以没有定格在更高难度的档次上，最后定格该题为C类难度。

### 4.练习与模拟

#### 通关考模拟卷

以提交系统设置“亲近通关考”和“模拟通关考”两个考项，供学生点击做题。亲近通关考提供标程和测试数据，可以对照着操作、琢磨、练习；模拟通关考不提供标程，可以在网上锻炼自己的真实做题能力。看到差距，努力弥合。同时公布电子稿，方便学生离线做题。

模拟通关考与亲近通关考与实际的通关考难度略有一点出入，实际通关考难度只会降低不会升高。

（见《模拟通关考》，《亲近通关考》）

#### 通关考解答代码

解答代码的题序由代码文件第一个字母表示，难度由最后一个字母表示。例如，第四题“无秤售油”（OilSail），其难度为B，则取名为DOilSail\_B.cpp。

解答代码只是方法相对典型、简捷的代码，并不是唯一解。可以用任何方法解答，只要能过题。因此，用C/C++语言解答都可以，只要最后能通过提交系统。

（见《亲近通关考解答代码》）

另外，还分类提供了通关考练习题集。

（见《通关练习题集》）

## 五．克服练习障碍

### 1.先过操作关

#### 须有实践机会和指导

通关考是课程教学环节的一部分，必须首先克服操作关。而这必须在任课老师的指导下完成。任课教师应在提交平台上安排数次课内实验。

#### 应掌握的操作内容

掌握在实验室电脑或自己电脑中如何打开编程平台和提交平台（包括注册登录）；

理解练习题（也是通关考题）描述的结构，能阅读理解题目；

学习掌握在电脑中如何创建一个程序工程，包括设置编程路径，建立合法工程文件夹；

学习掌握在编程平台中编程，编译、链接、运行、调试等步骤；

掌握将运行通过的代码复制黏贴到提交平台的对应处，并学会观察反馈信息。

#### 机器操作障碍

诸多鼠标无反应、键盘不起作用、网络不通、电脑死机等故障都不应作为阻扰学习编程的障碍。克服这些故障的经验应该在平时有所积累，在练习和考试中应有解决此类问题的直觉。

#### 提交操作障碍

能打开提交平台页面；

能正确登录用户名和密码；

能自如看到题库题和考试栏目中的题目；

能自如完成对应题目的代码提交（包括选择编译器）；

能查看提交反馈信息；

能正确对待提交时出现waiting…的反馈信息，不作无谓的重复提交。

### 2.认识提交信息

#### 1 Accepted提交成功

任何提交的练习者，都盼望着这个信息能够出现。只有唯一的一个红色字体信息Accepted，标志着“恭喜你答对了”。

#### 2 Presentation Error格式错误

格式错误指输出格式错误，一般是输出结果中，行末多了空格与回车。

题目设计（或者标程）的输出，其行末一般是没有多余的空格的。若提交的代码，运行输出的结果中，其行末含有空格或回车，就会发生格式错误。但是这种错误需要与行中间多了或少了空格要区别开。下面的表格说明了发生Presentation Error的情况以及与Wrong Answer错误的区别。

|  |  |  |  |
| --- | --- | --- | --- |
| 错误信息 | 标程输出 | 提交输出 | 说明 |
| Presentation Error | ABC  DEFG | ABC□  DEFG□ | 行末多一空格 |
| Presentation Error | ABC□  DEFG□ | ABC  DEFG | 行末少一空格 |
| Wrong Answer | 123□456  234□567 | 123456  234567 | 串或数中少空格 |
| Wrong Answer | 123456  234567 | 123□456  234□567 | 串或数中多空格 |
| Presentation Error | ABC  DEFG | ABC  DEFG | 行末多一回车 |
| Presentation Error | ABC  DEFG | ABC  DEFG | 行末少一回车 |

注：□表示空格。

发生格式错误一般是输出语句表示方式的错误。

例如希望输出：ABCABCABCABCABC

**for**(**int** i=0; i<5; i++)

{

cout<<”ABC\n”; // 每一个输出都多余了回车

**}**

cout<<”\n”; // 该语句若漏了，也会报告格式错误

又例如希望输出：0□1□2□3□4

**for**(**int** i=0; i<5; i++)

{

cout<<i<<” ”; // 输出0□1□2□3□4□ 多余了最后面的空格

**}**

cout<<”\n”;

应改为：

**for**(**int** i=0; i<4; i++) // 减少一次循环

{

cout<<i<<” ”; // 输出0□1□2□3□

**}**

cout<<”4\n”; // 补充输出最后一个4

这种错误是最容易查找的，因为一般不涉及算法正确性，只要仔细比照样本输出，检查输出语句是否有表示错误就行了。

#### 3 Compilation Error编译错误

提交系统有四种编译器可供选择：BCB6，VC8，G++，GCC。学生在课程学习中，每个教师可能会用不同的编译器环境来引导学生入门。因而，提交时需要考虑编译器的对应。如果提交时，选择了错误的编译器，就可能反馈编译错误的信息。

C++过程化编程中，也有标准化版本的些微差别。对于在自己的电脑上调试通过的程序，其按对应编译器提交一般不会有编译错误。很多时候，因为C++程序大体上是标准的，所以一个代码可能在任何一个编译器下都能运行，即不产生编译错误。这个时候，对于比较各个编译器下产生的代码之运行性能是很有助益的。

有些操作熟练的学生，发现了比较明确的代码错误，为了省却调试步骤，直接修改代码而提交，这时候，如果考虑欠周，则也可能出现编译错误。

使用C语言编程的学生，也可以在这四种编译器中的任何一种中提交，因为C++相当程度上兼容了C。不过偶尔也可能产生由语言兼容性所导致的编译错误。

发生了编译错误，只要将代码换到对应编译器上重新提交一次就行了。

编译错误的蓝色字体是表示提交平台Status选项下Result一栏中该错误信息可以点击进入查看具体的错误原因。由于自己电脑的编译调试功能很充分，所以这个点击查看功能通常用不到而处于退化状态。

#### 4 Wrong Answer结果错误

这种错误说明提交代码运行的结果与标准答案不符。

这种错误，一般说明提交的代码中含有逻辑（算法）错误，往往样本数据能够通过，但是系统中的测试数据未能通过。

这种错误，需要从问题描述理解的源头，重新仔细分析输入数据的表示范围和数据量。这个若没有分析正确，所编的代码必然就会有逻辑错误。例如：计算阶乘，输入数据描述为若干个整数，其表示范围在1到15之间：

样本输入数据列出

3 5

样本输出数据列出

6

120

若用int类型计算数值，能够保证自己电脑中样本数据的正确输入和输出。但明眼人会发现，阶乘值在求超13！时，int类型的表示将溢出。所以，这样的代码提交上去就有可能发生结果错误。可以用long long int类型代替int以获得正确计算结果。又假如该输入数据的表示范围在1到100之间，那么连long long int都不能表示了，要用大数方法来求解该问题。这时候，题目的难度也提升了，该题目应属于至少C+或者D类难度。

有了表示范围，在检查程序错误中，可以用一些非样本数据来调试代码，特别提倡用表示范围内的边界数据。例如，上面问题的输入，用15去运行试试，就能发现错误，以致找到原因。

问题描述中，一般不明确说明数据量，应该总是假定有成千上万个数据。所以，编程需要尽量用计算性能比较好的方法。方方面面的编程技巧，在《C++程序设计教程详解——过程化编程》中都有叙及，特别是慎用递归，这个需要学习相关的递归原理和编程注意事项。

#### 5 Time Limit Exceeded耗时超量（超时）错误

发生超时错误，源于两种情况：

1. 由于程序逻辑错误，对于特定的输入数据，使程序运行进入了高慢甚至死循环。
2. 强算法题，要求数据处理性能高；

前者就是编程逻辑问题了，需要检查边界数据在运行中的情况。

后者主要是由问题（题目）的难度决定的，其难度往往达到C+或D以上，多为竞赛题。所以，为了提高编程水平，高端的学生就需要了解算法。然而C++课程教学中，不涉及有难度的算法。特别是通关考，其题目要求的能力都是很有限的，不会超过C++课程学习范围，完全可以通过已有高中数学知识分析而得问题解决方案，设计计算表达的代码。

算法是一种积累，不能操之过急，许多学生到了二、三年级，才慢慢变成为ACM编程高手。

在通关考中，除非是做附加题，一般很少遇到超时错误。如果遇到超时错误，那发生的第一种情况比较多些，需要查看程序计算逻辑，看有没有出现死循环。

#### 6 Memory Limit Exceeted存储超量错误

发生存储超量错误，源于数组和其他数据集合的使用超量。这种超量，在自己电脑调试中，因为只运行样本数据而没有发现。例如，一次动态申请较大的空间以获得存储中间结果和计算速度。或者使用多次大量空间的数据集合。

这种情况发生时，一般要从两个方面去把握：

1. 看看能否优化数据集合的尺寸；
2. 将定义数据集合的语句从函数中搬到全局空间位置。因为全局数据空间的资源略优于栈数据区资源。

特别是递归结构的运行中，如果每个递归函数都要定义一个数量不小的数组，那么看都不用看，一定是这个原因了。

#### 7 Output Limit Exceeted输出超量错误

发生输出超量错误，往往是系统错误所引起，这种错误发生概率很小，而且一般不是个体问题，所有提交该题的代码都将遇到同样的错误。发生这种错误后，可以经过系统及时维护，而保证考试继续进行下去。

另一种可能是，输出语句的执行停不下来，导致输出量超出输出测试数据很多，还没有比较运行结果，就被输出超量错误给定性了。

#### 8 Run Time Error运行错误

这个运行错误，一般指程序中有逻辑错误。这种逻辑错误会导致计算机软件发生一个硬件中断错误，从而终止程序运行。激发硬件中断的原因有很多，但都不常见。常规的运行错误分四类：

1. 除0错。这个错，不是在编程中，写道“int a = b/0;”，而是类似：

**int** a, b, c;

cin>>b>>c;

a = b/c; // c的值是在运行中决定的，编程中没法知道

要避免这种错，请用if语句。

1. 数组下标溢出。例如：

**int** a[10];

**for**(**int** i=0; i<n; i++)

a[i] = 3; // 因为n>10而致错

**int** a[10];

**for**(**int** i=0; i<=10; i++)

a[i] = 3; // 因为访问a[10]而致错

1. string串重复度变负数

string是C++使用字串的资源，它使用方便，可以使程序简捷。但是当访问string串中某个字符的时候，因为涉及下标，可能会发生与第二种错误情形相同的错误。

string串可以表示一定重复度的相同字符的字串，例如：string(5, ’A’)表示字串“AAAAA”。更多的情况，是用一个整型变量表达式来表示重复度，从而代替一个for循环的作用。例如：cout<<string(n-m, ‘B’); 表示：

**for**(**int** i=0; i<n-m; i++)

cout<<’B’;

但是当n-m为负数时，for循环不会发生错误，大不了不输出，而string表示将上演一个运行错误。

1. 动态内存申请

在动态内存申请失败时，将给出运行错误。如果动态内存申请成功，但超过了内存使用限量，则报告存储超量错误。

#### 9 Unknown Error未知错误

提交的代码在运行中，由于输入测试数据的激发，导致运行出现异常。个别运行错误的处理是通过等待系统处理反馈一个结果而结束的，但是若这个等待超慢，就将演变为Unknown Error错误。

#### 10 关于waiting…

系统经常会反馈waiting…，表示“请你等待（程序还没运行好，结果还没出来）”。

因为网络拥堵，在偶尔达到1个提交/每秒的快速时，发生waiting是正常的。这时候的waiting…现象，只要通过刷屏就可以解除。Waiting 将考验系统的正常恢复能力，就像交通拥堵，能否快速疏通很重要，不然会让许多人误事。考试中发生网络拥堵，如果不能马上疏通，将影响学生的考试绝对用时，还影响考试的情绪。如果拥堵不能马上恢复，那么就要采取延长考试时间的措施。

发生网络拥堵，有两个原因：

1. 个别人有意无意的重复提交，虽然系统采用了屏蔽2秒内的两次同一地点发来的提交，但是，无意义的频繁提交显然不利于网络畅通。
2. 代码中含有让系统等待的直接或间接功能调用。这个一般是无意的，除非是高手。

发生waiting…现象，最好的应对是过几秒钟（不要太频繁）刷一次屏。观察一下，如果还是waiting…，并且其他提交也跟着waiting…，那就需要人为干预了，请尽快报告相关教师（若在考试中则报告监考教师）。

## 六．现场考试应对

### 1.正确操作用户名密码

对于考试开始时获得的用户名和密码，是唯一识别自己的身份和对应自己的考试成绩的。所以必须用该用户名和密码登录。

当编完代码提交时，切记不要在题库中提交，要在考试栏目（在cpp.zjut.edu.cn中点击contests可看到“某某场通关考”，继续点击进入）下对应某个考试题目（6个题之一），点击submit提交。这样的通关考操作，才能承认你的成绩（点击standing可看到排名）。

对于acm.zjut.edu.cn，操作类似。

### 2.做出题数即考试成绩

考试成绩基于考试排名中的做出题数。所以做出多少题的数量是硬道理。

### 3.题目选做策略

如果考生水平有限，应该尽量先做题库题，然后做难度低的题，最后做难度高的题。

然而，通关考与其他的程序设计竞赛一样，题目是不按难度顺序排列的，也不标出题目难度。

通关考是开卷的，每人手头上应该有题库题清单。这是查找和判断题库题的依据。

通关考是实时排名的，每个人当前已做出题数都在排名中显示。所以，不但可以看到自己做出了几题，也可以看到别人做出了几题。不但可以看到别人做出了哪些题，还可以看到哪个题通过率高。

判断难度低的题，甚至不需要看题目描述，只要看实时排名中哪个题通过率高即可。因此，跟风策略总是对的，也就是说，别人做哪题，我也做哪题，这样做题速度就快。

从心理上，先做容易题更容易接受。如果先做难题，当难题耗费了很多时间做出来后，再去做容易题，时间的紧迫性可能会影响做题速度。

### 4.代码风格不重要

代码风格是编写程序时，书写的格式习惯。通关考强调的是程序符合问题解决要求的正确性，提交时不要求代码风格，只要符合语法。能运行通过就是王道。

通关考的题目难度有限，代码量有限，所以，格式上不会阻碍程序编写的逻辑正确性。适当的键盘操作和鼠标操作熟练度，对于提高编程效率会有一些帮助。

### 5.做不出的应对策略

#### 想不出方法

原创题含有一定的知识集合。如果没有融会贯通这些知识点，即使分析出应该采用什么技术去解决，也可能实现不了（编不出程序）。

这是因为平时做题数量不够多，或者做题覆盖的知识面比较狭窄，或者学习方法不对。这就要从自己的学习方法上去找原因：化了时间在编程上了吗？会学习别人编程的方法吗？会看和理解书上的方法吗？能记下老师讲的方法吗？

考试中遇到这种情况，应回忆或者理一下自己曾做过哪些题，与正在做的题有哪些相似处，查阅相关的代码。这也是稳定情绪，保持正常发挥的一种心态。

#### 不能运行

特别是编好了程，上上下下看了个够，没有任何毛病，但就是不能运行。

1. 需要理一下程序能通过编译和链接吗？检查出现什么错，创建程序工程的方法是否和平时一样，路径设置是否对。
2. 检查一下输入数据和问题描述的输入数据结构是否一致。
3. 检查一下，这种机房环境的电脑是否第一次用，对系统的性能和表现是否熟悉。这种机器或许存在平时做题中的机器故障和软件故障。例如BCB软件的console输出窗口在许多机器上需要激活一下。

#### 提交失败

即使能运行，但提交就是通不过或者看不到排名变化。

1. 看一下提交反馈信息，根据反馈信息来判断和查找错误的原因（见前面的认识提交信息）。
2. 如果是最难查的Wrong Answer反馈信息，则应该重视题目中输入数据描述的理解。在自己电脑中先测试几个边界数据看看。
3. 确认一下自己的每个操作步骤是否正确。必要时需请监考老师确认运行环境的正常。

#### 调整心态

1. 换个题目试试：这题做不出，是哪里扳牢了，或许换个题目做做就好了；
2. 是平时做题做得不多，对能力培养的思路不够重视（心态放低，认识到自己的不足，有利于调整考试心态）；
3. 缺乏承受能力，缺少心理准备，相信能在下一场通关考中过关（吃一堑长一智，这本身就是通关考的设计原则）。

## 七．参考书籍

《C++程序设计教程（第二版）实验指导》 清华大学出版社 钱能 2007

《C++程序设计（第三版）》 清华大学出版社 谭浩强 2015

《C++程序设计教程详解——过程化编程》 清华大学出版社 钱能 2014

《C++程序设计教程（第二版）习题解答》 清华大学出版社 钱能 2009

《C++标准语言库》 华中科技大学出版社 侯捷 2009

《C++程序设计教程（修订版）——设计思想与实现》 清华大学出版社 钱能 2007

《C++程序设计教程（第二版）》 清华大学出版社 钱能 2005

2019年3月6日