[**ACM中Java的应用**](http://www.cppblog.com/vontroy/archive/2010/05/24/116233.html)

先说一下Java对于ACM的一些优点吧：  
(1) 对于熟悉C/C++的程序员来说Java 并不难学，两周时间基本可以搞定一般的编程，再用些时间了解一下Java库就行了。Java的语法和C++非常类似，可以说是C++的升级版，只是更加强调面向对象思想而已。（个人见解。。。）

(2) 在一般比赛中，Java程序会有额外的时间和空间，但真正进行大规模运算时Java并不比C/C++慢，输入输出效率比较低而已  
(3) Java 代码简单且功能强大，有些像高精度之类的算法用Java实现起来更为简洁方便（ACM真正比赛时是讲究做题速度的，任何题只要能过就行，而不必过于要求程序的速度有多高，不超时就好）。  
  
\*\*\*小技巧：某些题目用Java超时的话可以用Java打表然后用C/C++提交

(4) 用Java不易犯细微的错误，比如C/C++中的指针， “if (n = m) ... ” 等。

(5) 目前Eclipse已成基本配置，写Java程序反而比C/C++更方便调试。在具体竞赛时也算多一种选择。

关于ACM中应用的一些问题：  
(1) JDK 1.5.0 及其以上版本提供的Scanner类为输入提供了良好的基础，很好地优化Java的输入问题。  
     代码如下：

http://www.cppblog.com/Images/OutliningIndicators/None.gifimport java.io.\* import java.util.\*   
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifpublic class Main {   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    public static void main(String args[])   
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif         Scanner cin = new Scanner(new BufferedInputStream(System.in));       
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        http://www.cppblog.com/Images/dot.gif   
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }   
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

也可以直接 Scanner cin = new Scanner(System.in); 加Buffer可能会快一些。  
  
(2) 读一个整数：   int n = cin.nextInt();         相当于   scanf("%d", &n);   或 cin >> n;   
     读一个字符串：String s = cin.next();         相当于   scanf("%s", s);    或 cin >> s;   
     读一个浮点数：double t = cin.nextDouble();   相当于   scanf("%lf", &t); 或 cin >> t;   
     读一整行：     String s = cin.nextLine();     相当于   gets(s);  或 cin.getline(...);   
     判断是否有下一个输入可以用 cin.hasNext() 或 cin.hasNextInt() 或 cin.hasNextDouble()

(3) 输出一般可以直接用 System.out.print() 和 System.out.println()，前者不输出换行，而后者输出。

http://www.cppblog.com/Images/OutliningIndicators/None.gifSystem.out.println(n);   // n 为 int 型 同一行输出多个整数可以用   
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifSystem.out.println(new Integer(n).toString() + " " + new Integer(m).toString());   
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif//也可重新定义：   
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifstatic PrintWriter cout = new PrintWriter(new BufferedOutputStream(System.out));       
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gifcout.println(n);

(4)对于输出浮点数保留几位小数的问题，可以使用DecimalFormat类，

http://www.cppblog.com/Images/OutliningIndicators/None.gifimport java.text.\*;   
http://www.cppblog.com/Images/OutliningIndicators/None.gif    DecimalFormat f = new DecimalFormat("#.00#");   
http://www.cppblog.com/Images/OutliningIndicators/None.gif    DecimalFormat g = new DecimalFormat("0.000");   
http://www.cppblog.com/Images/OutliningIndicators/None.gif    double a = 123.45678, b = 0.12;   
http://www.cppblog.com/Images/OutliningIndicators/None.gif    System.out.println(f.format(a));   
http://www.cppblog.com/Images/OutliningIndicators/None.gif    System.out.println(f.format(b));   
http://www.cppblog.com/Images/OutliningIndicators/None.gif    System.out.println(g.format(b));

**大数：**

BigInteger 和 BigDecimal 是在java.math包中已有的类，前者表示整数，后者表示浮点数

http://www.cppblog.com/Images/OutliningIndicators/None.gifimport java.math.\*  // 需要引入 java.math 包   
http://www.cppblog.com/Images/OutliningIndicators/None.gifBigInteger a = BigInteger.valueOf(100);   
http://www.cppblog.com/Images/OutliningIndicators/None.gifBigInteger b = BigInteger.valueOf(50);   
http://www.cppblog.com/Images/OutliningIndicators/None.gifBigInteger c = a.add(b)   // c = a + b;   
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif//主要有以下方法可以使用：   
http://www.cppblog.com/Images/OutliningIndicators/None.gifBigInteger add(BigInteger other)   
http://www.cppblog.com/Images/OutliningIndicators/None.gifBigInteger subtract(BigInteger other)   
http://www.cppblog.com/Images/OutliningIndicators/None.gifBigInteger multiply(BigInteger other)   
http://www.cppblog.com/Images/OutliningIndicators/None.gifBigInteger divide(BigInteger other)   
http://www.cppblog.com/Images/OutliningIndicators/None.gifBigInteger mod(BigInteger other)   
http://www.cppblog.com/Images/OutliningIndicators/None.gifint compareTo(BigInteger other)   
http://www.cppblog.com/Images/OutliningIndicators/None.gifstatic BigInteger valueOf(long x)   
http://www.cppblog.com/Images/OutliningIndicators/None.gif  
http://www.cppblog.com/Images/OutliningIndicators/None.gif//输出数字时直接使用 System.out.println(a) 即可

**字符串：**

String 类用来存储字符串，可以用charAt方法来取出其中某一字节，计数从0开始：

String a = "Hello";    // a.charAt(1) = 'e'

用substring方法可得到子串，如上例

System.out.println(a.substring(0, 4))     // output "Hell"

注意第2个参数位置上的字符不包括进来。这样做使得 s.substring(a, b) 总是有 b-a个字符。

字符串连接可以直接用 + 号，如

String a = "Hello";   
String b = "world";   
System.out.println(a + ", " + b + "!");    // output "Hello, world!"

如想直接将字符串中的某字节改变，可以使用另外的StringBuffer类。

**调用递归（或其他动态方法）**

在主类中 main 方法必须是 public static void 的，在 main 中调用非static类时会有警告信息，   
可以先建立对象，然后通过对象调用方法：

http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifpublic class Main {   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        http://www.cppblog.com/Images/dot.gif   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    void dfs(int a)   
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif              if (http://www.cppblog.com/Images/dot.gif) return;   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        http://www.cppblog.com/Images/dot.gif   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif             dfs(a+1);   
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif    public static void main(String args[])   
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        http://www.cppblog.com/Images/dot.gif   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif         Main e = new Main();   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif              e.dfs(0);   
http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif        http://www.cppblog.com/Images/dot.gif   
http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }   
http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

**其他注意的事项：**

(1) Java 是面向对象的语言，思考方法需要变换一下，里面的函数统称为方法，不要搞错。

(2) Java 里的数组有些变动，多维数组的内部其实都是指针，所以Java不支持fill多维数组。   
数组定义后必须初始化，如 int[] a = new int[100];

(3) 布尔类型为 boolean，只有true和false二值，在 if (...) / while (...) 等语句的条件中必须为boolean类型。   
在C/C++中的 if (n % 2) ... 在Java中无法编译通过。

(4) 下面在java.util包里Arrays类的几个方法可替代C/C++里的memset、qsort/sort 和 bsearch:

Arrays.fill()   
Arrays.sort()   
Arrays.binarySearch()     
  
虽然Java功能很强大，但不能完全依赖他，毕竟C和C++还是ACM/ICPC的主流语言，适当地使用才能有效提高比赛中的成绩。。。  
  
附：  
例题：POJ 1001

 1http://www.cppblog.com/Images/OutliningIndicators/None.gifimport java.io.\*;  
 2http://www.cppblog.com/Images/OutliningIndicators/None.gifimport java.util.\*;  
 3http://www.cppblog.com/Images/OutliningIndicators/None.gifimport java.math.BigDecimal;  
 4http://www.cppblog.com/Images/OutliningIndicators/None.gif  
 5http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockStart.gifpublic class Main {  
 6http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif  
 7http://www.cppblog.com/Images/OutliningIndicators/InBlock.gifpublic static void main(String args[])  
 8http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif{  
 9http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif   Scanner cin = new Scanner(System.in);  
10http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif     
11http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif   BigDecimal num;  
12http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif   int n;  
13http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif   String r;  
14http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif     
15http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif   while(cin.hasNextBigDecimal())  
16http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif  {      
17http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif      num = cin.nextBigDecimal();  
18http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif     n = cin.nextInt();  
19http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif      num = num.pow(n);  
20http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif      r = num.stripTrailingZeros().toPlainString();  
21http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif     if(r.startsWith("0."))  r = r.substring(1);  
22http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif      
23http://www.cppblog.com/Images/OutliningIndicators/InBlock.gif       System.out.println(r);  
24http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif   }  
25http://www.cppblog.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif}  
26http://www.cppblog.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}  
27http://www.cppblog.com/Images/OutliningIndicators/None.gif