第3章 编程基础（II）

吴永辉

ACM-ICPC Asia Programming Contest 1st Training Committee – Chair

Shanghai Normal University，ACM-ICPC Asia Training League

yhwu@fudan.edu.cn

WeChat : 13817360465

目录

3.1 函数

- 结构化思想
  - 自顶向下
  - 逐步求精
  - 功能分解

- 函数，英语是“Function”，还有一个含义：功能。
- 一个函数的本质是在程序设计中，按模块化的原则，实现某一项功能。
- 程序是由一个主函数和若干个函数构成，主函数调用其他函数，其他函数之间也可以互相调用，并且一个函数可以被其他函数调用多次。

- //实验【3.1.1 Specialized Four-Digit Numbers】，【3.1.2 Pig-Latin】和【3.1.3 Tic Tac Toe】给出以函数实现特定功能。

- 函数定义的形式为：
  - “返回类型 函数名(形参列表){函数体语句 return 表达式;}”;
- 函数调用的形式为：
  - “函数名(实参列表);”;

3.1.1 Specialized Four-Digit Numbers

- 试题来源：**ACM Pacific Northwest 2004**
- 在线测试：**POJ 2196，ZOJ 2405，UVA 3199**

- 找到并列出所有具有这样特性的十进制的4位数字：其4位数字的和等于这个数字以16进制表示时的4位数字的和，也等于这个数字以12进制表示时的4位数字的和。
- 例如，整数2991的（十进制）4位数字之和是 2+9+9+1 = 21，因为 2991 = 1*1728 + 8*144 + 9*12 + 3，所以其12进制表示为1893_{12}，4位数字之和也是21。但是2991的十六进制表示为BAF_{16}，并且11+10+15 = 36，因此2991被程序排除了。
- 下一个数是2992，3种表示的各位数字之和都是22 (包括$BB0_{16}$)，因此2992要被列在输出中。（本题不考虑少于4位数字的十进制数----排除了前导零----因此2992是第一个正确答案。）

- 输入
- 本题没有输入。
- 输出
- 输出为2992和所有比2922大的满足需求的4位数字（以严格的递增序列），每个数字一行，数字前后不加空格，以行结束符结束。输出没有空行。输出的前几行如下所示。

试题解析

- 首先，设计一个函数*Calc*(*base*, *n*)，计算和返回*n*转换成*base*进制后的各位数字之和。
- 然后，枚举[2992..9999]内的每个数*i*：若*Calc*(10, *i*)== *Calc*(12, *i*)==*Calc*(16, *i*)，则输出*i*。

3.1.2 Pig-Latin

- 试题来源：**University of Notre Dame Local Contest 1995**
- 在线测试：**UVA 492**

- 您意识到PGP加密系统还不足够保护您的电子邮件，所以，你决定在使用PGP加密系统之前，先把您的明文字母转换成Pig Latin（一种隐语），以完善加密。

- 输入和输出
- 请您编写一个程序，输入任意数量行的文本，并以Pig Latin输出。每行文本将包含一个或多个单词。一个“单词”被定义为一个连续的字母序列（大写字母和/或小写字母）。单词根据以下的规则转换为Pig Latin，非单词的字符在输出时则和输入中出现的完全一样：
- [1] 以元音字母（a、e、i、o或u，以及这些字母的大写形式）开头的单词，要在其后面附加字符串“ay”（不包括双引号）。例如，“apple”变成“appleay”。
- [2] 以辅音字母（不是A, a, E, e, I, i, O, o, U 或 u的任何字母）开头的单词，要去掉第一个辅音字母，并将之附加在单词的末尾，然后再在单词的末尾加上“ay”。例如，“hello”变成“ellohay”。
- [3] 不要改变任何字母的大小写。

试题解析

- 首先，设计两个函数*isab*(char *c*)和*vowel*(char *c*)，分别判断字符*c*是否是字母，以及是否是元音字母。
- 在主程序中，在输入文本后，根据试题描述中给出的规则进行处理：非字母的字符，直接输出；如果是单词（一个连续的字母序列），则如果单词是辅音字母开头，则把该辅音字母放到单词的最后；然后，所有的单词后加上“ay”。

3.1.3 Tic Tac Toe

- 在线测试：**POJ 2361，ZOJ 1908，UVA 10363**
- 试题来源：**Waterloo local 2002.09.21**

三连棋游戏（Tic Tac Toe）是一个在 3×3 的网格上玩的少儿游戏。一个玩家 X 开始将一个‘X’放置在一个未被占据的网格位置上，然后另外一个玩家 O，则将一个‘O’放置在一个未被占据的网格位置上。‘X’和‘O’就这样被交替地放置，直到所有的网格被占满，或者有一个玩家的符号在网格中占据了一整行（垂直，水平，或对角）。

初始的时候，用 9 个点表示为空的三连棋，在任何时候放‘X’或放‘O’都会被放置在适当的位置上。图 3.1-1 说明了从开始到结束三连棋的下棋步骤，最终玩家 X 获胜。

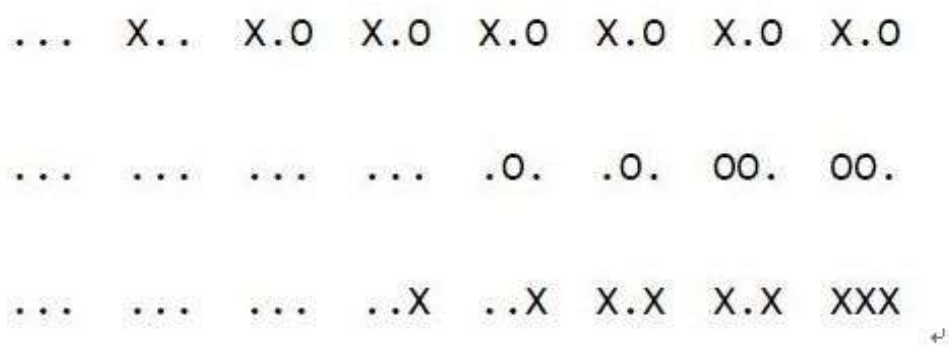


图 3.1-1

请您编写一个程序，输入网格，确定其是否是有效的三连棋游戏的一个步骤？也就是说，通过一系列的步骤可以在游戏的开始到结束之间产生这一网格？

- 输入
- 输入的第一行给出*N*，表示测试用例的数目。然后给出4*N*-1行，说明*N*个用空行分隔的网格图。
- 输出
- 对于每个测试用例，在一行中输出"yes"或"no"，表示该网格图是否是有效的三连棋游戏的一个步骤。

试题解析

由于玩家 X 先走且轮流执子，因此网格图为有效的三连棋游戏的一个步骤，一定同时呈现下述特征：

1. ‘O’的数目一定小于等于‘X’的数目；
2. 如果‘X’的数目比‘O’多 1 个，那么不可能是玩家 O 赢了三连棋；
3. 如果‘X’的数目和‘O’的数目相等，则不可能玩家 X 赢了三连棋。

也就是说，网格图为无效的三连棋游戏的一个步骤，至少呈现下述 5 个特征之一

1. ‘O’的个数大于‘X’的个数
2. ‘X’的个数至少比‘O’的个数多 2
3. 已经判出玩家 O 和玩家 X 同时赢
4. 已经判出玩家 O 赢，但‘O’的个数与‘X’的个数不等；
5. 已经判出玩家 X 赢，但双方棋子个数相同；

否则网格图为有效的三连棋游戏的一个步骤。

- 全局变量
  - 整个程序都可以使用的变量。
- 局部变量
  - 只能在局部使用的变量，也就是说，只能在特定的函数或子程序中可以访问的变量，它的作用域就在函数或子程序的内部。

3.1.4 Factorial! You Must be Kidding!!!

- 试题来源：**The Conclusive Contest- The decider. 2002**
- 在线测试：**UVA 10323**

Arif 在 Bongobazar 买了台超级电脑。Bongobazar 是达卡（Dhaka）的二手货市场，因此他买的这台超级电脑也是二手货，存在一些问题。其中的一个问题是这台电脑的 C/C++编译器的无符号长整数的范围已经被改变了。现在新的下限是 10000，上限是 6227020800。Arif 用 C/C++写了一个程序，确定一个整数的阶乘。整数的阶乘递归定义为：

$$\text{Factorial}(0) = 1$$

$$\text{Factorial}(n) = n*\text{Factorial}(n-1).$$

当然，可以改变这样的表达式，例如，可以写成：

$$\text{Factorial}(n) = n*(n-1)*\text{Factorial}(n-2)$$

这一定义也可以转换为迭代的形式。

但 Arif 知道，在这台超级电脑上，这一程序不可能正确地运行。请您编写一个程序，模拟在正常的计算机上的改变行为。

- 输入
- 输入包含若干行，每行给出一个整数*n*。不会有整数超过6位。输入以EOF结束。
- 输出
- 对于每一行的输入，输出一行。如果*n*!的值在Arif 计算机的无符号长整数范围内，输出行给出*n*!的值；否则输出行给出如下两行之一：
- Overflow! //(当 *n*! > 6227020800)
- Underflow! //(当 *n*! < 10000)

试题解析

本题题意非常简单：给出 n，如果 $n!$ 大于 6227020800，则输出"Overflow!"；如果 $n!$ 小于 10000，输出"Underflow!"；否则，输出 $n!$。

$F(n)=n*F(n-1)$，并且 $F(0)=1$。虽然负阶乘通常未被定义，但本题在这一方面做了延伸：$F(0)=0*F(-1)$，即 $F(-1)=\frac{F(0)}{0}=\infty$。则 $F(-1)=-1*F(-2)$，也就是 $F(-2)=-F(-1)=-\infty$。以此类推，$F(-2)=-2*F(-3)$，则 $F(-3)=\infty$……

- 首先，离线计算$F[i]=i!$，$8\le i\le 13$。
- 然后，对每个n，
  - 如果$8\le n\le 13$，则输出$F[n]$；
  - 如果($n\ge 14$||($n<0$&&(-n)%2==1))，则输出"Overflow!"；
  - 如果($n\le 7$ || ($n<0$&&(-n)%2==0))，则输出"Underflow!"。
- 在参考程序中，函数*init*()离线计算$F[i]=i!$，$0\le i\le 13$。在函数中，循环变量i为局部变量；而$F[i]$则为全局变量。

3.2 递归函数

程序调用自身的编程技巧称为递归（Recursion），是子程序在其定义或说明中直接或间接调用自身的一种方法。

例如，对于自然数 n，阶乘 $n!$的递归定义为 $n!=\begin{cases}1 & n=0\\ n\times(n-1)! & n\geq 1\end{cases}$。则按阶乘 $n!$的递归定义，求解 $n!$的递归函数 *fac*(n)如下。

```
int   fac(int n);
{
      if (n==0) return 1;                  //判断递归边界
      if (n>=1) return n*fac(n-1);         //处理递归并返回结果
}
```

从上述例子可以得出：首先，基于递归的定义给出递归函数；其次要有递归的边界条件（结束条件）；而递归的过程是向递归的边界条件不断逼近。

3.2.1 Function Run Fun

- 试题来源：**ACM Pacific Northwest 1999**
- 在线测试：**POJ 1579**

- 我们都爱递归！不是吗？
- 请考虑一个三参数的递归函数$w(a, b, c)$：
  - if $a \le 0$ or $b \le 0$ or $c \le 0$, then $w(a, b, c)$ 返回1；
  - if $a > 20$ or $b > 20$ or $c > 20$, then $w(a, b, c)$ 返回$w(20, 20, 20)$；
  - if $a < b$ and $b < c$, then $w(a, b, c)$ 返回$w(a, b, c-1)+w(a, b-1, c-1)- w(a, b-1, c)$；
  - 否则，返回$w(a-1, b, c) + w(a-1, b-1, c) + w(a-1, b, c-1) - w(a-1, b-1, c-1)$。
- 这是一个很容易实现的函数。但存在的问题是，如果直接实现，对于取中间值的a、b和c（例如，a=15，b=15，c=15），由于大量递归，程序运行非常耗时。

- 输入
- 程序的输入是一系列整数三元组，每行一个，一直到结束标志-1-1-1为止。请您高效地计算$w(a, b, c)$并输出结果。
- 输出
- 输出每个三元组$w(a, b, c)$的值。

试题解析

- 对于取中间值的a、b和c，由于大量递归，程序运行非常耗时。所以，本题的递归函数计算采用记忆化递归进行计算，用一个三维的数组f来记忆递归的结果，$f[a][b][c]$用于记忆$w(a, b, c)$的返回值。

3.2.2 Simple Addition

- 试题来源：**Warming up for Warmups, 2006**
- 在线测试：**UVA 10944**

定义一个递归函数 $F(n)$，其中，

$$F(n)=\begin{cases} n\%10 & \text{若}(n\%10)>0 \\ 0 & \text{若}n=0 \\ F(n/10) & \text{其他} \end{cases}；$$

定义另一个函数 $S(p,q)$，$S(p,q)=\sum_{i=p}^{q}F(i)$。

给出 p 和 q 的值，求函数 $S(p,q)$的值

- 输入
- 输入包含若干行。每行给出两个非负整数p和q（$p \le q$），这两个整数之间用一个空格隔开，p和q是32位有符号整数。输入由包含两个负整数的行结束。程序不用处理这一行。
- 输出
- 对于每行输入，输出一行，给出$S(p, q)$的值。

试题解析

- 根据递归函数$F(n)$的定义给出递归函数。因为p和q是32位有符号整数，$S(p, q)$的值可能会超出32位有符号整数，所以本题的变量类型定义为long long int，即64位有符号整数。

因为 $q-p$ 可能高达 2^{31}，如果直接按题意，对于 $p \le n \le q$ 的每个 n，计算 $F(n)$并累加到 $S(p, q)$中，可能会导致程序超时。因此，要根据 p 和 q 之间的范围，分而治之地进行优化：

如果 $q-p<9$，根据 $S(p,q)=\sum_{i=p}^{q} F(i)$，直接递归计算每个 $F(i)$，并累加计算 $S(p, q)$。

如果 $q-p \ge 9$，则分析递归函数 $F(n)$：如果 $n\%10 \ne 0$，即 n 的个位数不为 0，则 $F(n)=n\%10$，即 n 的个位数。因此，对于个位数不为 0 的 n，将其个位数计入总和。如果 $n\%10=0$，则 $F(n)=F(n/10)$，即，在个位数为 0 时，就要将其十位数变为个位数，进行上述分析。

- 本题递归算法：
  - 每一轮求出[*p*, *q*]区间内数字的个位数的和，并计入总和；再将[*p*, *q*]区间内个位数为0的数除以10，产生新区间，进入下一轮，再求新区间内数字的个位数的和，并计入总和，而个位数为0的数除以10，产生新区间；以此类推；直到[*p*, *q*]区间内的数只有个位数。

- 例如，求$S(2, 53)$，将范围划分为3个区间：[2, 9]，[10, 50]和[51, 53]。
- 对于第1个区间[2, 9]，个位数之和2+3+4+……+9=44；对于第2个区间[10, 50]，个位数之和(1+2+……+9)×4= 45×4 = 180；对于第3个区间[51, 53]，个位数之和1+2+3 = 6。所以，第一轮，个位数的总和为44+180+6 = 230。
- 在[10, 50]中，10, 20, 30, 40和50的个位数是0，将这些数除以10后得1, 2, 3, 4, 5，产生新区间[1, 5]；进入第二轮，区间[1, 5]中的数只有个位数，个位数之和1+2+3+4+5=15。
- 最后，两轮结果相加，得$S(2, 53)$=230+15=245。

3.3 结构体

- 在C语言中，可以使用结构体（Struct）将一组不同类型的数据组合在一起。
- 结构体的定义形式为：“struct 结构体名{ 结构体所包含的变量或数组 };”。

3.3.1 A Contesting Decision

- 试题来源：**ACM Mid-Atlantic 2003**
- 在线测试地址：**POJ 1581，ZOJ 1764，UVA 2832**

- 对程序设计竞赛进行裁判是一项艰苦的工作，要面对要求严格的参赛选手，要作出乏味的决定，以及要进行着单调的工作。不过，这其中也可以有很多的乐趣。
- 对于程序设计竞赛的裁判来说，用软件使得评测过程自动化是一个很大的帮助，而一些比赛软件存在的不可靠也得使人们希望比赛软件能够更好、更可用。您是竞赛管理软件开发团队中的一员。基于模块化设计原则，您所开发模块的功能是为参加程序设计竞赛的队伍计算分数并确定冠军。给出参赛队伍在比赛中的情况，确定比赛的冠军。

- 记分规则：
- 一支参赛队的记分由两个部分组成：第一部分是被解出的题数，第二部分是罚时，表示解题总的耗费时间和试题没有被解出前错误的提交所另加的罚时。对于每个被正确解出的试题，罚时等于该问题被解出的时间加上每次错误提交的20分钟罚时。在问题没有被解出前不加罚时。
- 因此，如果一支队伍在比赛20分钟的时候在第二次提交解出第1题，他们的罚时是40分种。如果他们提交第2题3次，但没有解决这个问题，则没有罚时。如果他们在120分钟提交第3题，并一次解出的话，该题的罚时是120分。这样，该队的成绩是罚时160分，解出了两道试题。
- 冠军队是解出最多试题的队伍。如果两队在解题数上打成平手，那么罚时少的队是冠军队。

- 输入
- 程序评判的程序设计竞赛有4题。本题设定，在计算罚时后，不会导致队与队之间不分胜负的情况。
- 第1行 为参赛队数*n*
- 第2 - *n*+1行为每个队的参赛情况。每行的格式
- < *Name* > < *p*1*Sub* > < *p*1*Time* > < *p2Sub* > < *p2Time* > ... < *p*4*Time* >
- 第一个元素是不含空格的队名。后面是对于4道试题的解题情况（该队对这一试题的提交次数和正确解出该题的时间（都是整数））。如果没有解出该题，则解题时间为0。如果一道试题被解出，提交次数至少是一次。
- 输出
- 输出一行。给出优胜队的队名，解出题目的数量，以及罚时。

试题解析

- 本题的参考程序用结构体表示参赛队信息，结构体team_info中包含参赛队名、4道题提交次数、4道题解题时间、解题数，以及总罚时。所有的参赛队则表示为一个结构体数组*team*。

- 设冠军队的队名为*wname*，解题数为*wsol*，罚时为*wpt*。
- 首先，依次读入每个队的队名*name*和4道题的提交次数*subi*，解题时间*timei*；并计算每个队的解题数和总罚时。
- 然后，依次处理完*n*个参赛队的信息，若当前队解题数最多，或虽同为目前最高解题数但罚时最少（(team[i].num>wsol) || (team[i].num==wsol && team[i].time<wpt)），则将当前队暂设为冠军队，记下队名、解题数和罚时。
- 在处理完*n*个参赛队的信息后，*wname*、*wsol*和*wpt*就是问题的解。

- 日期由年、月、日来表示，平面坐标系由X坐标和Y坐标表示，所以日期类型、坐标类型可以用结构体来表示。
- 实验【**3.3.2 Maya Calendar**】和实验【**3.3.3 Diplomatic License**】分别用结构体存储日期类型和坐标类型。

3.3.2 Maya Calendar

- 试题来源：**ACM Central Europe 1995**
- 在线测试：**POJ 1008，UVA 300**

- 上周末，M. A. Ya教授对古老的玛雅有了一个重大发现。从一个古老的节绳（玛雅人用于记事的工具）中，教授发现玛雅人使用Haab历法，一年有365天。Haab历法每年有19个月，在前18个月，每月有20天，月份的名字分别是pop, no, zip, zotz, tzec, xul, yoxkin, mol, chen, yax, zac, ceh, mac, kankin, muan, pax, koyab, cumhu。这些月份中的日期用0到19表示；Haab历的最后一个月叫做uayet，它只有5天，用0到4表示。玛雅人认为这个日期最少的月份是不吉利的：在这个月，法庭不开庭，人们不从事交易，甚至不打扫房屋。

- 因为宗教的原因，玛雅人还使用了另一个历法，这个历法中年被称为Tzolkin历法（holly年），一年被分成13个不同的时期，每个时期有20天，每一天用一个数字和一个单词相组合的形式来表示。使用的数字是1~13，使用的单词共有20个，它们分别是：imix, ik, akbal, kan, chicchan, cimi, manik, lamat, muluk, ok, chuen, eb, ben, ix, mem, cib, caban, eznab, canac, ahau。注意：年中的每一天都有着明确唯一的描述，比如，在一年的开始，日期如下描述： 1 imix, 2 ik, 3 akbal, 4 kan, 5 chicchan, 6 cimi, 7 manik, 8 lamat, 9 muluk, 10 ok, 11 chuen, 12 eb, 13 ben, 1 ix, 2 mem, 3 cib, 4 caban, 5 eznab, 6 canac, 7 ahau, 8 imix, 9 ik, 10 akbal ……也就是说数字和单词各自独立循环使用。

- Haab历和Tzolkin历中的年都用数字0，1，……表示，数字0表示世界的开始。所以第一天被表示成：
- Haab: 0. pop 0
- Tzolkin: 1 imix 0
- 请您帮助M. A. Ya教授，编写一个程序，把Haab历转化成Tzolkin历。

- 输入
- Haab历中的数据由如下的方式表示：
- *NumberOfTheDay*. *Month Year*（日期. 月份 年数）
- 输入中的第一行表示要转化的Haab历日期的数据量。接下来的每一行表示一个日期，年数小于5000。
- 输出
- Tzolkin历中的数据由如下的方式表示：
- *Number NameOfTheDay Year*（天数字 天名称 年数）
- 第一行表示输出的日期数量。下面的每一行表示一个输入数据中对应的Tzolkin历中的日期。

试题解析

- 在参考程序中，Haab历和Tzolkin历的月份，分别用字符串数组*haab*和*tzolkin*表示；而日期类型由年、月、日组成，用结构体*data*表示。
- 设Haab历的日期为*year*年*month*月*date*天，则这一日期从世界开始计起的天数*current*。
- 对于第*current*天来说，Tzolkin历的日期为*year*年的第*num*个时期内的第*word*天。由于Tzolkin历每年有260天（13个时期，每时期20天），因此若*current* % 260=0，则表明该天是Tzolkin历中某年最后一天，即*year*=*current*/260-1，*num*=13，*word*=20天；若*current* % 260≠0，则*year*=*current*/260；*num*=(*current* % 13 == 0 ? 13 : *current* % 13)，*word*=(*current* - 1) % 20+1。

3.3.3 Diplomatic License

- 试题来源：**Ulm Local 2002**
- 在线测试：**POJ 1939**

- 为了尽量减少外交开支，世界各国讨论如下。每一个国家最多只与一个国家保持外交关系是不够的，因为世界上有两个以上的国家，有些国家不能通过（一连串的）外交官进行相互交流。
- 本题设定每个国家最多与另外两个国家保持外交关系。平等对待每个国家是一条不成文的外交惯例。因此，每个国家都与另外两个国家保持外交关系。
- 国际地形学家提出一种适合这一需求的结构。他们将安排国家组成一个圈，使得每个国家都与其左右两个邻国建立外交关系。在现实世界中，一个国家的外交部是设在这个国家的首都。为了简单起见，本题设定，首都的位置是二维平面上的一个点。如果您用直线把保持外交关系的相关国家的外交部联起来，结果就是一个多边形。
- 现在，要为两个国家之间的双边外交会议设定地点。同样，出于外交原因，两国的外交官前往该地点的距离必须相等。为了提高效率，应尽量缩短行驶距离，请您为双边外交会议做好准备。

- 输入
- 输入给出若干测试用例。每个测试用例首先给出数字n，表示涉及n个国家。本题设定$n \geq 3$是一个奇数。然后，给出n对x和y坐标，表示外交部的位置。外交部的坐标是绝对值小于10^{12}的整数。国家的排列顺序与它们在输入中出现的顺序相同。此外，在列表中，第一个国家是最后一个国家的邻国。
- 输出
- 对于每个测试用例，首先输出测试用例中国家的数量（$=n$），然后给出国家之间的双边外交会议地点位置的x和y坐标。输出的会议地点的顺序应与输入给出的顺序相同。从排在最前的两个国家的会议地点开始，一直到排在最后面的两个国家的会议地点，最后输出第n个国家和第一个国家的会议地点。

- 提示：国家之间组成一个圈可以被视为一个多边形。

试题解析

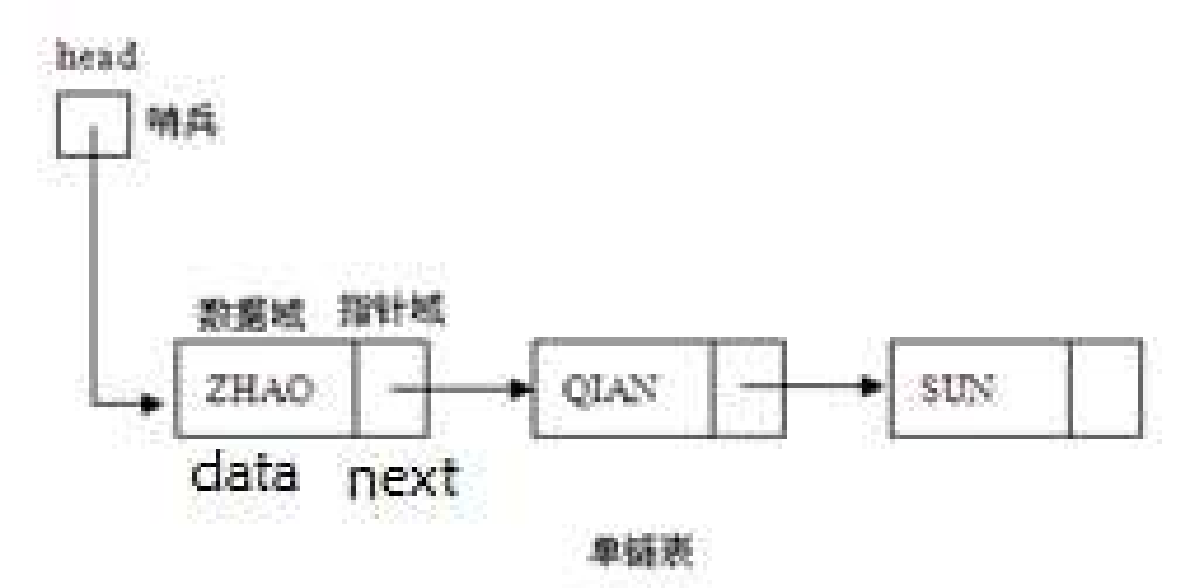
- 本题给出*n*个点的坐标，这*n*个点围城一个多边形，求这个多边形的*n*条边的中点坐标。最后一个中点坐标是输入的起点和终点的中点坐标。
- 用结构表示点的x和y坐标，由中点坐标公式给出两个相邻点的中点坐标。

3.4 指针

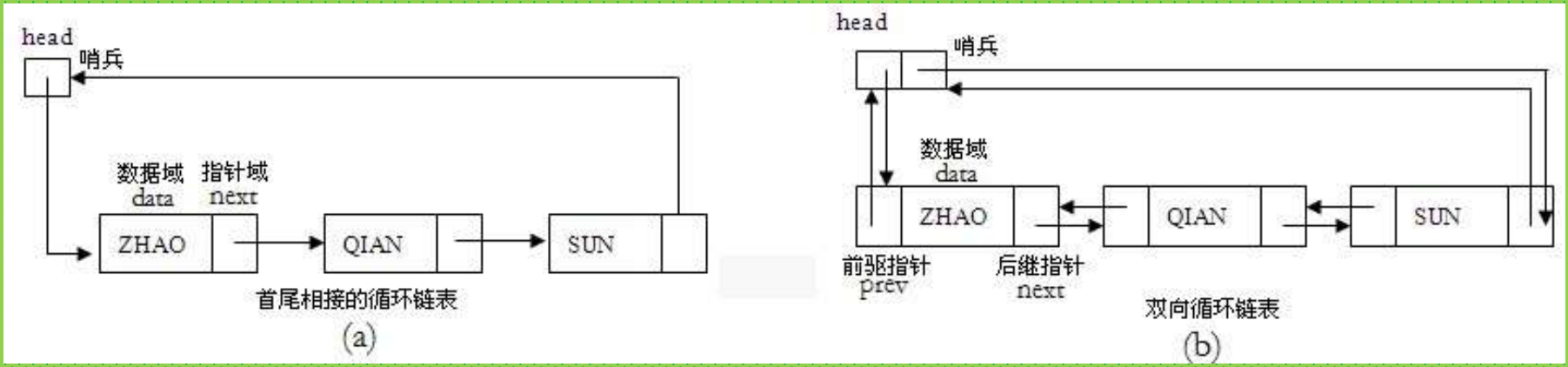
- 在程序设计语言中，指针是指内存地址，指针变量是用来存放内存地址的变量。
- 实验【**3.4.1 "Accordian" Patience**】是指针和结构体结合，构成线性表的链接存储结构。

- 线性表，就是通常所说的表格，是由相同类型的数据元素组成的有限、有序的集合，其特点是：
  - 1）线性表中元素的个数是有限的；
  - 2）线性表中元素是一个接一个有序排列的，除第一个数据元素外，每个数据元素都有一个前驱；除最后一个数据元素外，每个数据元素都有一个后继；
  - 3）所有的数据元素类型相同；
  - 4）线性表可以是空表，即表中没有数据元素。
- 线性表的链接存储结构，以结构体表示数据元素的类型，以指针将数据元素一个接一个地链接起来。

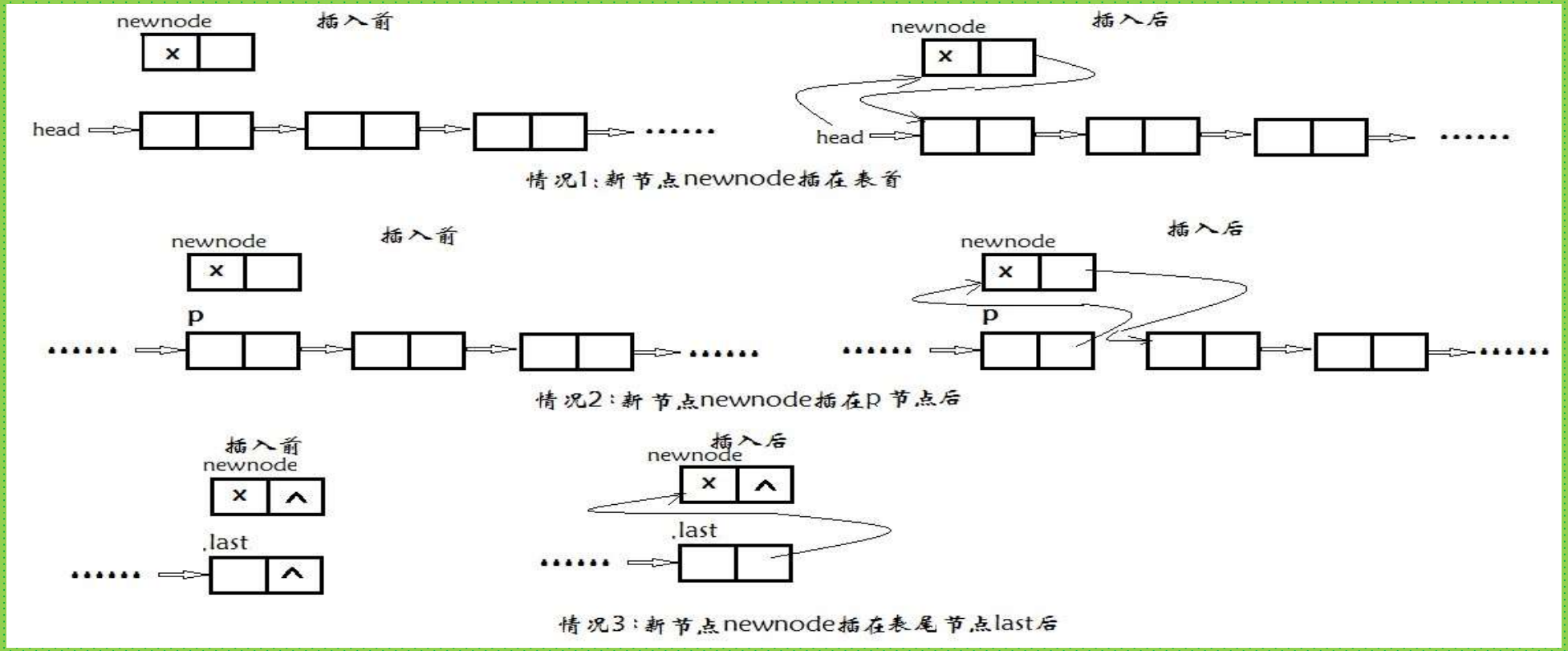
- 单链表（singly linked list）是一种最简单和最典型的链表。
  - 单链表有一个指示链表开始地址的表头（*head*），简称为哨兵；
  - 表中每个数据元素占用一个节点（*node*），节点类型为一个结构体，含两个域：一个域为数据域（*data*），其数据类型取决于数据对象的属性；另一个域为后继指针域（*next*），给出下一个节点的存储地址。



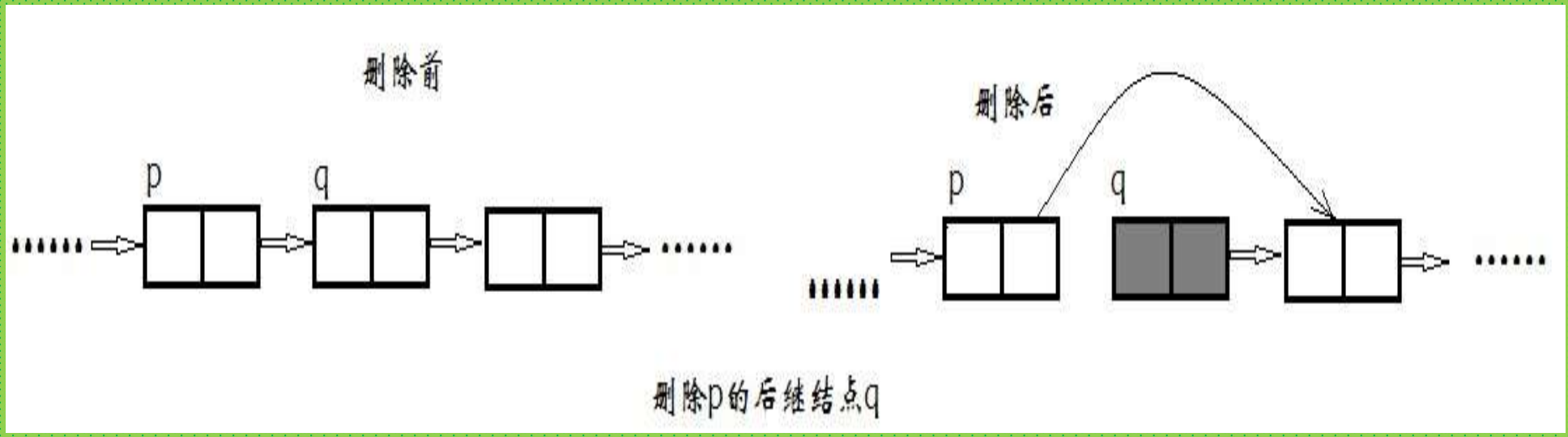
- 如果将最后一个节点的后继指针域指向哨兵，则单链表变成了循环链表（如图(a)）；
- 如果在循环链表的基础上，每个节点再增加一个指向直接前驱的*prev*指针，则循环链表又变成了双向循环链表(图(b))。



- 在单链表中插入一个地址为*newnode*的新节点，只要修改链中节点的后继指针值，无需移动表中的数据元素。
- 下图分别给出了新节点插入表首、中间位置和表尾的三种情况：



- 在单链表中删除一个节点的操作比较简单。若被删节点的地址为*q*，其前驱节点的地址为*p*，只要将*p*节点的后继指针指向*q*的后继节点地址，就可以使得地址为*q*的节点从链中分离出来，达到删除它的目的(如下图)。



3.4.1 "Accordian" Patience

- 试题来源：**New Zealand 1989**
- 在线测试：**UVA 127，POJ 1214**

- 请您模拟"Accordian" Patience游戏，规则如下：
- 玩家将一副扑克牌一张一张地发牌，从左到右排成一排，不能重叠。只要一张扑克牌和左边的第一张牌或左边的第三张牌相匹配，就将这张扑克牌移被匹配的牌的上面。所谓两张牌匹配是这两张牌的数值（数字或字母）相同或花色相同。每当移了一张牌之后，就再检查看这张牌能否继续往左移，每次只能移在牌堆顶部的牌。本游戏可以将两个牌堆变成一个牌堆，如果根据规则，可以将右侧牌堆的牌一张一张地移到左侧牌堆，就可以变成一个牌堆。本游戏尽可能地把牌往左边移动。如果最后只有一个牌堆，玩家就赢了。
- 在游戏过程中，玩家可能会遇上一次可以有多种选择的情况。当两张牌都可以被移动时，就移动最左边的牌。如果一张牌可以向左移动一个位置或向左移动三个位置，则将其移动三个位置。

- 输入
- 输入给出发牌的顺序。每个测试用例由一对行组成，每行给出26张牌，由单个空格字符分隔。输入文件的最后一行给出一个#作为其第一个字符。每张扑克牌用两个字符表示。第一个字符是面值（A=Ace，2-9，T=10，J=Jack，Q=Queen，K=King），第二个字符是花色（C= Clubs（梅花），D=Diamonds（方块），H=Hearts（红心），S=Spades（黑桃））。
- 输出
- 对于输入中的每一对行（一副扑克牌的52张牌），输出一行，给出在对应的输入行进行游戏后，每一堆扑克牌中剩余的扑克牌的数量。

试题解析

- 本题给一副扑克牌，一共52张。首先，将扑克牌从左往右一张张地排列。然后从左往右遍历，如果该牌和左边第一张牌或左边第三张牌相匹配，那么就将这张牌移到被匹配的牌上，形成牌堆；每次只能移动每堆牌最上面一张牌。两张牌匹配条件是面值相同或者花色相同。每次移动一张牌后，还应检查牌堆，看有没有其他牌能往左移动；如果没有，遍历下一张牌，直到不能移动牌为止。最后，输出每一堆扑克牌中剩余的扑克牌的数量。

- 在参考程序中，扑克牌用带指针变量的结构体表示，其中两个字符变量*a*和*b*分别表示扑克牌的面值和花色，指针变量*pre*和*post*分别指向从左往右的顺序中的前一张牌和后一张牌，而指针变量*down*则指向所在牌堆的下一张牌。这副扑克牌表示为一个三相链表，每一个牌堆用线性链表表示，而在牌堆顶部的牌，其*pre*和*post*分别指向前一个牌堆顶部的牌和后一个牌堆顶部的牌。
- 本题根据题目给定的规则，模拟发牌和移动牌的过程。这里要注意,根据题意，应先比较左边第三张，然后，再比较左边第一张。

- 在参考程序中，由于对线性链表的操作函数频繁地调用，所以，相关的函数被声明为内联函数（inline）。

- 在实验【**3.4.2 Broken Keyboard (a.k.a. Beiju Text)**】的参考程序中，使用数组模拟线性链表。

3.4.2 Broken Keyboard (a.k.a. Beiju Text)

- 试题来源：**Rujia Liu's Present 3: A Data Structure Contest Celebrating the 100th Anniversary of Tsinghua University**
- 在线测试：**UVA 11988**

- 您正在用一个坏键盘键入一个长文本。这个键盘的问题是时不时“Home”键或“End”键会在您输入文本时自动按下。您并没有意识到这个问题，因为你只关注文本，甚至没有打开显示器。完成键入后，您打开显示器，在屏幕上看到文本。在中文里，我们称之为悲剧。请您是找到悲剧的文本。

- 输入
- 输入给出若干测试用例。每个测试用例都是一行，包含至少一个，最多100,000个字母、下划线和两个特殊字符‘[’和‘]’；其中‘[’表示“Home”键，而‘]’表示“End”键。输入以EOF结束。
- 输出
- 对于每个测试用例，输出在屏幕上的悲剧的文本。

试题解析

- 本题题意：
  - 对于每个输入的字符串，如果出现'['，则输入光标就跳到字符串的最前面，如果出现']'，则输入光标就跳到字符串的最后面。输出实际上显示在屏幕上的字符串。

- 将输入的字符串表示为链表，再输出。其中，每个字符为链表中的元素的数据，而指针指向按序输出的下一个元素。
- 用数组模拟链表链表：用数组*next*代替链表中的*next*指针，例如，第一个字符*s*[1]的下一个字符是*s*[2]，则*next*[1]=2。此外，对于链表，第0个元素不储存数据，作为辅助头结点，第一个元素开始储存数据。
- 设置变量*cur*表示光标，*cur*不是当前遍历到的位置*i*，是表示位置*i*的字符应该插入在*cur*的右侧。如果当前字符为'['，光标*cur*就跳到字符串的最前面，即*cur*=0；如果']'，光标就跳到字符串的最后面，即*cur=last*，其中变量*last*保存当前字符串最右端的下标。
- 程序根据试题描述给出的规则进行模拟。