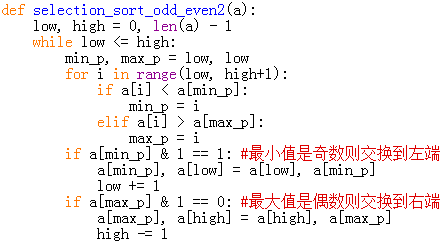
Python算法之旅（第22期）

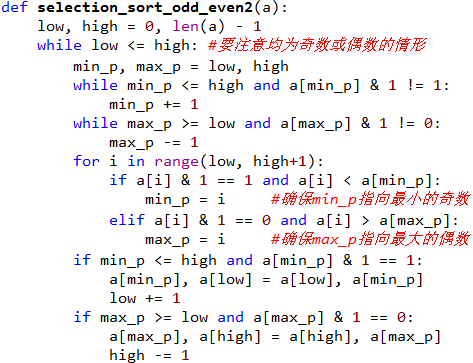
上期回顾：

描述：有一组正整数，要求分别对奇数和偶数进行升序排序，其中奇数在前，偶数在后。我们给出了一段具有迷惑性的错误代码：



它的主要问题在于扫描过程中没有注意最值下标的奇偶性，一旦出现最小值是偶数，且最大值是奇数的情形时，就会陷入死循环。

改进后的代码如下：



排序是高考算法题中永恒的主题，也是所有计算机软件中最重要的算法，生活中很多实用的程序都需要用到排序算法，接下来我们就来分析一个实用的程序，欣赏其中巧妙的排序算法。

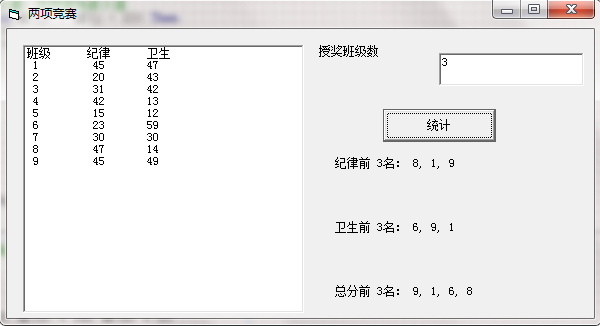
题目：计算两项竞赛的名次和总分

难度：4星 有趣：4星 有用：4星

分类：排序， 模拟

描述：某学校使用“两项竞赛”对各班级的日常表现进行评比，规则如下：记录各个班级的纪律和卫生得分，根据得分分别计算纪律和卫生名次（得分越高排名越前，得分相同，名次也相同）；再根据两项竞赛的名次计算总分（总分=纪律名次+卫生名次）；最后计算总分的名次（总分越低排名越前，得分相同，名次相同）。现用程序实现上述功能：窗体加载时，自动从数据库读取各班的纪律和卫生得分，显示在列表框List1中；在文本框Text1中输入授奖班级数n，单击“统计”按钮Command1，在标签Label1、Label2和Label3中分别输出纪律、卫生、总分前n名的班级编号。

程序运行界面如图所示。



（1）根据评比规则和图中数据，班级1两项竟赛的总分是\_\_\_\_\_\_\_\_

（2）实现上述功能的VB程序如下，请在划线处填人合适代码。

Const n = 9

'数组bj、df和mc分别存储班级编号、得分和名次；各数组的1~n元素存储纪律相关数据；n+1~2n元素存储卫生相关数据；2n+1~3n元素存放两项竞赛总分相关数据

Dim bj(1 To n \* 3) As Integer, df(1 To n \* 3) As Integer, mc(1 To n \* 3) As Integer

Dim i As Integer, j As Integer, t As Integer

'本过程从数据库读取各班级的班级编号、得分数据，分别存储在数组bj、df中

Private Sub Form\_Load()

'代码略

End Sub

Private Sub Command1\_Click()

Dim c As Integer

c = Val(Text1.Text)

Label1.Caption = "纪律前" & Str(c) & "名：" & pm(1, n, c, False)

Label2.Caption = "卫生前" & Str(c) & "名：" & pm(n + 1, 2 \* n, c, False)

For i = 1 To n

bj(2 \* n + i) = i: df(2 \* n + i) = 0: mc(2 \* n + i) = 0

Next i

For i = 1 To 2 \* n

df(2 \* n + bj(i)) = df(2 \* n + bj(i)) + mc(i)

Next i

Label3.Caption = "总分前" & Str(c) & "名：" & pm( ① , c, True)

End Sub

’函数pm实现对指定区间的数组元素进行排序，并返回前若干名的班级编号。ks和js分别表示该区间的起止位置；num表示返回的班级数；fx表示排序方向，true 表示升序，false表示降序

Function pm(ks As Integer, js As Integer, num As Integer, fx As Boolean) As String

For i = ks To js - 1

For j = js To ks + 1 Step -1

If (fx And df(j) < df(j - 1)) Or ( ② ) Then

t = df(j): df(j) = df(j - 1): df(j - 1) = t

t = bj(j): bj(j) = bj(j - 1): bj(j - 1) = t

End If

Next j

Next i

mc(ks) = 1: pm = Str(bj(ks))

For i = ks + 1 To js

If df(i) = df(i - 1) Then

③

Else

mc(i) = i - ks + 1

End If

If mc(i) <= num Then pm = pm & "," & Str(bj(i))

Next i

End Function

答案：

（1）5

（2）① 2 \* n + 1, 3 \* n

② Not fx And df(j) > df(j - 1)

③ mc(i) = mc(i - 1)

算法分析：

分析本题时有三个地方需要注意：

1.对数组bj、df和mc三个不同区间元素的理解：

它们分别存储班级编号、得分和名次；各数组的1~n元素存储纪律相关数据；n+1~2n元素存储卫生相关数据；2n+1~3n元素存放两项竞赛总分相关数据。

题目中Form\_Load()过程的代码省略了，需要我们自行脑补缺失的代码。因为各数组2n+1~3n元素在Command1\_Click()过程中进行了设置，故我们只需初始化1~2n区间的元素。根据程序运行界面，我们为Form\_Load()过程补充代码如下：

Private Sub Form\_Load()

List1.Clear

List1.AddItem "班级" + " " + "纪律" + " " + "卫生"

For i = 1 To n

bj(i) = i: bj(n + i) = i

Next i

df(1) = 45: df(2) = 20: df(3) = 31: df(4) = 42: df(5) = 15: df(6) = 23

df(7) = 30: df(8) = 47: df(9) = 45

df(10) = 47: df(11) = 43: df(12) = 42: df(13) = 13: df(14) = 12: df(15) = 59

df(16) = 30: df(17) = 14: df(18) = 49

For i = 1 To n

List1.AddItem Str(bj(i)) + " " + Str(df(i)) + " " + Str(df(n + i))

Next i

End Sub

2. 对函数pm()的理解：

该函数做了三件事情，先对[ks,js]区间内的得分和对应班级进行排序，然后根据排序后的得分计算名次，并存储到数组mc中，最后输出前c名的班级序号bj(i) 。

本题采用了经典的冒泡排序，以得分为关键字对数组df和bj同时进行了排序，使得二者一直保持一一对应关系，即bj(i)的得分恰好是df(i)。

计算名次的算法采用如下逻辑：单项得分越高排名越前（总分越低排名越前），得分相同，名次也相同。所以第一名永远都是下标为ks的元素，即mc(ks) = 1，接下来从第二名开始计算，即For i = ks + 1 To js，其中i为元素的下标，若其得分与前一个元素不同，则其对应的名次为i - ks + 1，即mc(i) = i - ks + 1；否则它的名次和前一个元素相同，即mc(i) = mc(i - 1)。

3.如何计算和存储总得分：

总分=纪律名次+卫生名次，2n+1~3n元素存放两项竞赛总分相关数据。

Command1\_Click()过程做了四件事，先分别调用函数pm()输出了单项得分的前c个班级，然后对区间2n+1~3n的元素进行了初始化处理，再根据各班的单项排名情况，计算出各班的总分，最后调用函数pm()输出了总分的前c个班级。

其中计算班级总分的代码较难理解，代码中对df(2 \* n + bj(i))的处理类似桶排序算法。因为班级bj(i)的单项排名恰好对应mc(i)的值，所以可以通过对mc(i)求和来计算班级bj(i)的总分。

需要注意的是以下一段具有迷惑性的代码：

For i = 1 To n

df(2 \* n + bj(i)) = df(2 \* n + bj(i)) + mc(i) + mc(n + i)

Next i

此段代码误以为班级bj(i)的两个单项名次分别是mc(i)和mc(n + i)，想通过对它们求和得到总分。事实上bj(i)对应的纪律名次确实是mc(i)，但卫生名次却不一定是mc(n + i)，只有bj(n + i)对应的卫生名次才是mc(n + i)。所以我们可以把原代码改成：

For i = 1 To n

df(2 \* n + bj(i)) = df(2 \* n + bj(i)) + mc(i)

df(2 \* n + bj(n + i)) = df(2 \* n + bj(n + i)) + mc(n + i)

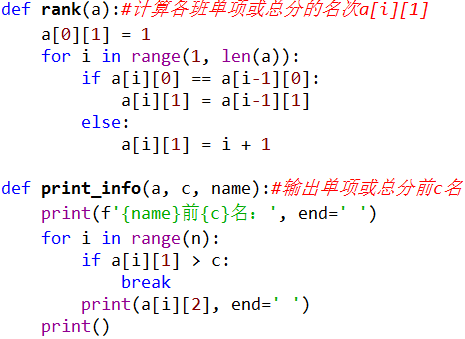
Next i

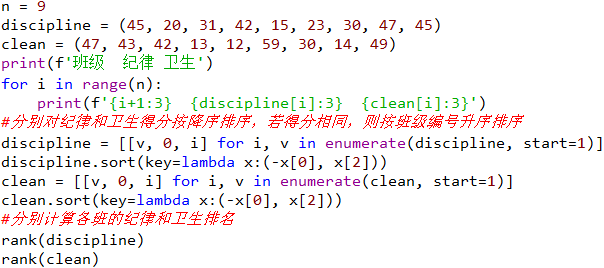
这样做更能体现桶排序的特征，但不如原来的代码简洁。

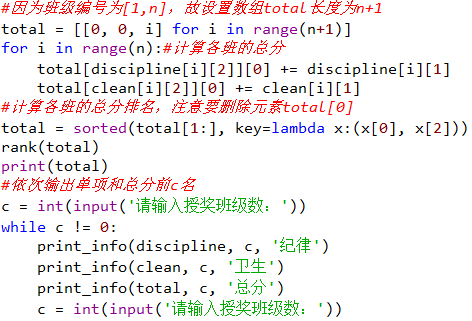
本题代码实现的功能非常实用，算法逻辑也很清晰，但是由于题目把不同单项的得分（或排名）存储在同一个数组的不同区间，处理数据的时候不够直观，增大了理解的难度。如果使用不同的数组来存储不同的项目，可以提高代码的可读性（虽然代码长度也增加了）。

我们利用Python语言灵活的数据结构和丰富的内置函数，可以在提高代码可读性的同时，保证代码足够简洁。在下面的代码中，我们设置了3个二维列表discipline，clean和total分别存储纪律，卫生和总分信息，其中discipline [i][0]，discipline [i][1]和discipline [i][2]分别代表第(i+1)个班级的得分，名次和班级编号，其他两个列表的数据结构也一样。

我们利用列表生成式来创建上述3个二维列表，然后依次对它们排序，再生成名次信息即可。具体代码如下：







课后思考：

在上述代码中，我们利用语句clean.sort(key=lambda x:(-x[0], x[2]))，实现了对列表clean进行多重排序的功能，主要关键字是clean [i][0]，次要关键字是clean [i][2]，其中对主要关键字按降序排序，次要关键字按升序排序。

因为列表clean的元素值都是数值，所以这样做是可行的。如果列表的元素值包含字符串，又该怎么处理呢？

描述：原列表a = [('aa','bb'),('cc','ee'),('bb','cc'),('aa','cc'),('bb','dd')]，现在要求对a进行排序，主要关键字是a[i][0]，次要关键字是a[i][1]，其中对主要关键字按升序排序，次要关键字按降序排序。

例如对于上述的列表a，排序后为a = [('aa', 'cc'), ('aa', 'bb'), ('bb', 'dd'), ('bb', 'cc'), ('cc', 'ee')]

聪明的你不妨想一想，我们下期公布答案。

另外，如果你有更 Pythonic（优雅的、地道的、整洁的）代码，或者与本文不同的算法思路和代码实现，请你一定留言或联系我，让我们一起讨论，共同进步。