```
1         Program f_radix;
2         {
            Задача.  Из текстового файла (не менее 900 Kb) прочитать
                     20 000 слов,
                     записать их в двумерный массива T (100, 200).
                     Отсортировать массив стрингов T методом RADIX.

            Решение. Сергей Митрофанов,
                     Центр НИТ "Северная Звезда",
                     17.11.03, 16:52-20:00
                     18.11.03, 09:40
          }

12        Uses
13          Dos,
14          Crt;

15        Const
16          p1 = 128; { код 'A' }
17          p2 = 159; { код 'Я' }
18          NR = 20; { число разрядов, такова длина слов в файле F }

19          { размер матрицы T (N, L) }
20          {
            N = 3;
            L = 4;
            }

24          N = 60; { строк }
25          L = 200; { столбцов}
26          PATH = 'D:\BP\BIN\STUD\';

27        Type
28          { массив под строку динамического массива }
29          S = array [1..L] of string [20];


30        Var
31          ch1, ch2, { час }
32          m1, m2, { минута }
33          s1, s2, { секунда }
34          sds1, sds2 { сотая доля секунды }
35          : word;

36          A, { массив указателей (адресов) на строки массива }
37          B  { вспомогательный массив, в нем сортировка по предыдущему
               разряду}
39          : array [1..N] of ^S;

40          t, { номер разряда-литеры слова }
```

```
41          d  { длина считанного слова }
42          : integer;

43          F, { данный текстовый файл }
44          R  { отсортированный файл }
45          : text;

46          i, j { номер элемента псевдо-линейного массива B }
47          : integer;

48          str, { строка данного файла }
49          slovo { слово, без конечных пробелов }
50          : string [20];


51        Function i_stroki (
52                             index { индекс одномерного }
53                             : integer
54                           )
55                           : integer;
56        {
            перевод индекса одномерного массива в индекс строки двумерного
          }

59        begin
60          if index mod L = 0
61            then
62              i_stroki := index div L
63            else
64              i_stroki := index div L + 1;
65        end;


66        Function i_stolb (
67                             index { индекс одномерного }
68                             : integer
69                           )
70                           : integer;
71        {
            перевод индекса одномерного массива в индекс столбца двумерного
          }

74        begin
75          if index mod L = 0
76            then
77              i_stolb := L
78            else
79              i_stolb := index mod L;
80        end;
```

```
81      Procedure count_sort;
82      {
          Сортировка подсчетом
          --------------------

          Этот метод подходит для сортировки целых чисел из не очень
          большого
          диапазона (сравнимого с размером массива).

          Идея вот в чем: для каждого элемента найти, сколько элементов,
          меньше определенного числа, и поместить это число на
          соответствующее место. Делается это так: за линейный проход по
          массиву мы для каждого из возможных значений подсчитываем,
          сколько элементов имеют такое значение. Потом добавляем к
          каждому из найденных чисел сумму всех предыдущих.
          Получая, таким образом, сколько есть элементов, значения которых
          не больше данного значения. Далее, опять-таки за линейный
          проход, формируем из исходного массива новый отсортированный.
          При этом следим, чтобы два одинаковых элемента не были записаны
          в одно место.
        }

100     var
101       C { частота букв разряда }
102       : array [p1 - 1..p2] of integer;

103       j, { номер элемента одномерного массива }
104       hj, vj, { координаты двумерного массива }
105       hf, vf, { координаты двумерного массива }
106       f { код символа слова }
107       : integer;

108       sym { символ слова }
109       : char;

110     begin
111       { обнулим массив C }
112       for j := p1 - 1 to p2 do
113         C [j] := 0;

114       {
            Узнаем, сколько раз для данного разряда t встречается та или
            иная цифра. Какова частота появления той или иной цифры в
            массиве чисел A
            t in [1..1000] - столько чисел в массиве A может быть
            Если t = 1 - младший разряд - разряд единиц

            Итак, за линейный проход по массиву мы для каждого из
            возможных значений (0..9) подсчитываем, сколько элементов
            имеют такое значение.
          }
```

```
124         for j := 1 to N * L do
125           begin
126             hj := i_stroki (j);
127             vj := i_stolb  (j);
128             sym := A [hj]^ [vj] [t];
129             if sym = ' '
130               then
131                 f := p1 - 1
132               else
133                 f := Ord (sym); { код символа слова на позиции t }
134             {
                f := (A [j] mod (t * 10)) div t;
                }
137             C [f] := C [f] + 1;
138           end;

139         {
              заметьте, не от 0, а от единицы! Почему?
              В каждом следующем элементе будет сумма двух предыдущих
              C [0] - останется неизменным
              а в C [9] - скопится сумма всех цифр разрядов.

              Итак, к каждому из найденных чисел добавлена сумма всех
              найденных предыдущих.

              В каждой C [i] будет будет записана информация о том, сколько
              чисел меньше i-того разряда. Этим самым мы будем знать место
              числа в массиве, который сортируем. Вот это место и будет
              записано в массиве C.
            }
151         for j := p1 to p2 do
152            C [j] := C [j - 1] + C [j];

153         for j := N * L downto 1 do
154           begin
155             hj := i_stroki (j);
156             vj := i_stolb  (j);

157             sym := A [hj]^ [vj] [t];
158             if sym = ' '
159               then
160                 f := p1 - 1
161               else
162                 f := Ord (sym); { код символа слова на позиции t }
163             {
                f := (A [j] mod (t * 10)) div t;
                }
166             hf := i_stroki (C [f]);
167             vf := i_stolb  (C [f]);
168             B [hf]^ [vf] := A [hj]^ [vj];
169             {
```

```
            B [C [f]] := A [j];
            }
172         C [f] := C [f] - 1;
173       end;
174     end; { end count_sort }


175     Begin
176       ClrScr;

177       WriteLn ('Сортирую текстовый файл методом RADIX');
178       WriteLn ('Подождите, пожалуйста...');

179       GetTime (ch1, m1, s1, sds1);
180       WriteLn ('Начало счета: ', ch1, ':', m1, ':', s1, ':', sds1);

181       {
          Assign (F, PATH + 'abc.txt');
          }
184       Assign (F, PATH + 'dwswords.txt');
185       Reset (F);

186       {
            получим N адресов начал адресов строк массивов A и B
          }
189       for i := 1 to N do
190         begin
191           New (A [i]);
192           New (B [i]);
193         end;

194       {
            Заполним массив A, читая данный файл F
          }
197       i := 1; { индекс 1 строки массива A }
198       j := 0; { индекс столбца A }
199       while not Eof (F) do
200         begin
201           ReadLn (F, str);
202           d := Length (str);
203           if d < NR
204             then
205               begin
206                 while d <> NR do
207                   begin
208                     str := str + ' ';
209                     Inc (d);
210                   end;
211               end
212             else
213               begin
```

```
214                 WriteLn ('Ошибка в файле...');
215                 WriteLn ('Нашлось слово, длинее NR = ', NR);
216                 ReadLn;

217                 Exit;
218               end;

219           Inc (j);
220           A [i]^ [j] := str;
221           if j = L
222             then
223               begin
224                 j := 0;
225                 Inc (i);
226               end;
227           if i > N
228             then
229               Break;
230         end;
231       Close (F);

232       {
            печать массива A
          }
235       {
          for i := 1 to N do
            begin
              for j := 1 to L do
                Write (A [i]^ [j]:NR, ' ');
              WriteLn;
            end;
          }

243       {
            отсортируем динамический двумерный массив строк
            методом RADIX.
          }
247       { t - номер литеры в слове }
248       for t := NR downto 1 do
249         begin
250           count_sort;
251           { перепишем элементы двумерного массива B в A }
252           for i := 1 to N do
253             for j := 1 to L do
254               A [i]^ [j] := B [i]^ [j];
255         end;

256       {
            печать массива A
          }
259       {
```

```
          WriteLn;
          WriteLn ('Отсортированный: ');
          for i := 1 to N do
            begin
              for j := 1 to L do
                Write (A [i]^ [j]:NR, ' ');
              WriteLn;
            end;
          }
269       {
          ReadLn;
          Exit;
          }

273       Assign (R, PATH + 'radix.txt');
274       Rewrite (R);

275       for i := 1 to N do
276         begin
277           for j := 1 to L do
278             begin
279               { удалим конечные пробелы }
280               str := A [i]^ [j];
281               slovo := '';
282               for t := 1 to Length (str) do
283                 if str [t] <> ' '
284                   then
285                     slovo := slovo + str [t]
286                   else
287                     Break;
288               { запишем слово без конечных пробелов в файл }
289               WriteLn (R, slovo);
290             end;
291           Dispose (A [i]);
292           Dispose (B [i]);
293         end;
294       Close (R);

295       WriteLn ('Отсортированный файл записан в "radix.txt"');

296       Sound (300);
297       Delay (3000);
298       NoSound;

299       WriteLn ('Начало счета: ', ch1, ':', m1, ':', s1, ':', sds1);

300       GetTime (ch2, m2, s2, sds2);

301       WriteLn ('Окончание счета: ', ch2, ':', m2, ':', s2, ':', sds2);

302       Readln;
```

```
303         End.
```

Listing данной задачи опубликован в сети Internet по адресу
http://www.Best-Listing.ru/color-1-task-584.html

Sergey Mitrofanov, 28.08.14, 19:43
E-mail: infostar@mail.ru