Гончаренко Валерий Евстафиевич
Доцент кафедры «Информационные технологии в экономике и организация производства» (ИТЭ и ОП) ГОУ ВПО «Ивановский государственный университет», кандидат технических наук. Ответственный организатор городской олимпиады школьников по информатике, член экспертной комиссии ЕГЭ по информатике и ИКТ.

Как применить сортировку подсчётом к действительным числам?

В № 1 за 2010 год была опубликована статья В.Е. Гончаренко «Алгоритм сортировки подсчётом и варианты его программной реализации», в которой рассматривалась проблема сортировки линейных массивов. Она вызвала большой интерес у наших читателей. И здесь мы рассмотрим, как можно использовать описанный алгоритм для сортировки не только целых, но и действительных чисел.

Большинство алгоритмов сортировки линейных массивов или иных линейных структур основываются на сравнениях и перестановках элементов, последовательно приближаясь к их упорядоченности. Количество этих операций в алгоритмах сортировки выбором или пузырьком гораздо больше, чем в каком-либо другом известном алгоритме сортировки, поэтому эти алгоритмы и характеризуются самой большой трудоёмкостью. Их вычислительная сложность $O(n^2)$, где n – количество элементов массива. Даже прославленный рекурсивный алгоритм быстрой сортировки не гнушается операциями сравнения и перестановок, но умудряется делать их намного меньше. Его вычислительная сложность $O(n \log_2 n)$. Чем больше n, тем ощутимее его преимущество по этому показателю.

В предлагаемом в статье алгоритме сортировки подсчётом не используются промежуточные перестановки элементов массива. Для них сразу определяется место в упорядоченной последовательности на числовой оси. Для одинаковых значений элементов подсчитывается, сколько раз они будут выстраиваться друг за другом, начиная с указанного места. Поэтому алгоритм и получил такое название. Его вычислительная сложность $O(n+k)$, где n – размер сортируемого массива, k –

размер вспомогательного массива, в котором ведётся подсчёт различных значений исходного массива, включая и отсутствующие. Значение k зависит от диапазона значений исходного массива и оценивается вариационным размахом $V = \max - \min$, где max – максимальное, а min – минимальное значение элементов массива.

Некоторые наши читатели оказались особенно придирчивыми. У них возникло замечание: «Ваш алгоритм сортирует только целые числа, действительные числа ему не по зубам!». Как это ни печально, но необходимо признать справедливость данного замечания. Его поддержат и все читатели, которые следят за массой своего тела, пользуясь замерами в долях кг. Врачи настоятельно рекомендуют это делать. Каждый грамм заслуживает пристального внимания, ведь 120,000 кг все-таки лучше, чем 120,001 кг.

В приведённом примере есть обнадёживающее обстоятельство – никому не приходит в голову контролировать массу тела с точностью до миллиграмма. Аналогичная ситуация может быть и во многих других примерах. Это обстоятельство вселяет надежду на возможность использования алгоритма сортировки подсчётом для действительных чисел, которые временно можно превратить в целые числа.

Метаморфоза действительных чисел

Не все числовые данные являются целыми числами. В связи с этим можно сделать вывод о невозможности использования алгоритма сортировки подсчётом для действительных (вещественных) чисел. В этом алгоритме значениям, которые надо упорядочить, ставится в соответствие номер элемента во вспомогательном массиве или записи в типизированном файле, а номер может быть только целым числом.

В подавляющем количестве задач в области экономики, технологий различных отраслей промышленности и др. обработка данных, представленных действительными числами, ведётся с округлением от одного до трёх знаков после десятичной запятой. Например, при расчёте затрат на горючее в авиапере-

возках пассажиров учитывается среднее значение массы тела пассажиров в кг. В этом случае нет практического смысла в учёте массы тела i-го пассажира с точностью более 0,001 кг, то есть с точностью большей, чем до третьего знака после десятичной запятой. Следовательно, действительные числа можно временно преобразовать в целые числа с учётом от одного до трёх и более знаков в дробной части числа.

В языке Pascal для преобразования вещественного типа в целый используются функции **round**(x) или **trunc**(x), где x – переменная вещественного типа. С учётом данных обстоятельств значения элементов линейного массива можно преобразовать к целому типу, умножая их на 10^k и преобразовывая полученные значения к целому типу с округлением, где k – количество знаков после запятой. В дальнейшем преобразованные значения восстанавливаются делением на 10^k с приведением к действительному типу.

Ниже приводится листинг программы, которая выполняет сортировку линейного массива действительных чисел из N элементов. Помимо файла f, с помощью которого выполняется сортировка массива, программа использует ещё один вспомогательный файл (fv), в который записываются преобразованные к целому типу N элементов исходного массива.

```
1   Program sort_real;
2   Uses Crt;
3   Const N=30;
4            Sort='c:\sort_re.sss';
5   Type Tmas=array[0..(N-1)] of real;
6   Var mas:Tmas;
7       fv:file of longint;
8       f:file of longint;
9       i,j,k,m:longint;
10      z:longint;
11      min,max,zn,r:longint;
12  begin
13       Clrscr;
14       randomize;
15       writeln('Исходный массив:');
16       for i:=0 to (N-1) do
17           begin
18                mas[i]:=100*random;
19                write(mas[i]:1:5,'  ');
20           end;
21       writeln;
22       assign(fv,'c:\sort_v.sss');
23       rewrite(fv);
24       writeln('Введите точность представления чи-
         сел(количество знаков ');
25       write('после запятой от 1 до 3) ');
26       readln(r);
27       zn:=1;
```

```
28        for i:=1 to r do
29            zn:=zn*10;
30        min:=round(mas[0]*zn);
31        max:=min;
32        for i:=0 to (N-1) do
33            begin
34                 z:=round(mas[i]*zn);
35                 write(fv,z);
36                 if z<min then min:=z;
37                 if z>max then max:=z;
38            end;
39        assign(f,Sort);
40        rewrite(f);
41        m:=0;
42        for i:=0 to (max-min) do
43            write(f,m);
44        seek(fv,0);
45        repeat
46              read(fv,z);
47              z:=z-min;
48              seek(f,z);
49              read(f,k);
50              inc(k);
51              seek(f,z);
52              write(f,k);
53        until Eof(fv);
54        j:=-1;
55        i:=-1;
56        seek(f,0);
57        repeat
58              inc(i);
59              read(f,k);
60              for m:=1 to k do
61                          begin
62                               inc(j);
63                               mas[j]:=(i+min)/zn;
64                          end;
65        until Eof(f);
66        writeln('Упорядоченный массив:');
67        for i:=0 to (N-1) do
68            write(mas[i]:1:r,'  ');
69        close(fv);
70        erase(fv);
71        close(f);
72        erase(f);
73        writeln;
74        writeln('Для завершения нажмите ENTER');
75        readln
76  End.
```

Комментарии к программе.

В строках 15–20 выполняется инициализация массива и вывод его на экран с округлением значений до 5-ти знаков после запятой.

В строках 22–23 объявляется вспомогательный файл *fv*, который открывается для чтения и записи.

В строках 24–29 пользователь вводит с клавиатуры точность представления действительных чисел от 1 до 3 знаков после запятой, что запоминается в переменной *r*, а затем в цикле **for** рассчитывается коэффициент $zn = 10^r$. Этот коэффициент используется для преобразования значений элементов массива в тип целого числа.

В строках 30–38 в цикле **for** преобразованные значения элементов массива записываются во вспомогательный файл *fv*, а также определяются среди этих значений наименьшее и наибольшее значения, которые хранятся соответственно в переменных min и max.

В строках 39–43 объявляется и открывается для чтения и записи файл *f*, в который затем записываются (max – min) + 1 нулевых значений.

В строках 44–53 из вспомогательного файла *fv* в цикле **repeat** последовательно считываются преобразованные значения элементов массива в переменную *z*, которая затем уменьшается на величину min. В файле *f* внутренний указатель смещается к записи с номером *z* и её значение считывается в переменную *k*, которая увеличивается на 1 и опять записывается в файл *f* на своё место. По своей сути *k* отражает количество элементов с одинаковым значением, соответствующим номеру записи в файле.

В строках 54–65 в цикле **repeat** считываются последовательно значения записей файла *f* в переменную *k*. Во внутреннем цикле **for** при изменении счётчика итераций от 1 до *k* выполняется генерация очередного номера элемента массива, которому присваивается преобразованное в обратную строку значение действительного типа.

В строках 67–68 на экран выводится упорядоченный по признаку неубывания значений линейный массив с точностью до *r* знаков после запятой.

В строках 69–72 файлы *f* и *fv* закрываются и удаляются.

В итоге программа выполнила сортировку линейного массива действительных чисел, округляя их по желанию пользователя от 1 до 3 знаков после запятой (этот диапазон можно и увеличить).

Преимущество данного алгоритма проявляется в большей мере при сортировке больших линейных структур, поэтому выбрана программная реализация его на примере линейной структуры записей в файле данных. Всё более увеличивающиеся ресурсы памяти ПК создают предпочтительные условия для применения алгоритма сортировки подсчётом.