Сортировка массивов

Общие сведения о сортировке

Сортировкой или упорядочением массива называется расположение его элементов по возрастанию (или убыванию).

Сортировка числовых и нечисловых данных – одна из важнейших процедур обработки информации, т.к. она существенно ускоряет последующий поиск объектов в массиве.

Если не все элементы различны, то надо говорить о неубывающем (или невозрастающем) порядке.

В обычном "неотсортированном" массиве информацию найти можно, но только способом последовательного перебора-просмотра всех его элементов, до тех пор, пока нужный элемент не будет обнаружен.

Этот метод требует много времени на каждый поиск. Все остальные методы поиска информации намного эффективнее, быстрее, но применяются только в упорядоченных, отсортированных массивах.

О значимости сортировки говорит тот факт, что в монографии Д.Кнута «Искусство программирования» (в 1999 г. включена в список двенадцати лучших физико-математических монографий столетия) **том 3 посвящен сортировке и поиску – 840 стр.**

Алгоритмов сортировки на сегодняшний день немало. Среди них есть простые и понятные, но неэффективные для больших массивов, например:

- **метод пузырька**
- **метод выбора**

Многие алгоритмы эффективны, но сложны для программной реализации (особенно для начинающих программистов). К ним относятся:

- **«быстрая сортировка» (*Quick Sort*)**

- **сортировка «кучей» (*Heap Sort*)**
- **сортировка слиянием**
- **пирамидальная сортировка**

На данном занятии будут рассмотрены два наиболее популярных простейших метода сортировки.

Сортировка методом пузырька (обменами)

> Идея метода состоит в сравнении двух соседних элементов, в результате чего меньшее число (более легкий "пузырек") перемещается на одну позицию влево.

Обычно просмотр организуют с конца, и после первого прохода самое маленькое число перемещается на первое место. Затем все повторяется от конца массива до второго элемента и т.д.

> При использовании этого алгоритма весь массив просматривается несколько раз подряд.

При каждом таком просмотре сравниваются последовательно только соседние элементы массива. Если при сравнении окажется, что предыдущий элемент больше следующего, они меняются местами описанным выше способом. Так в результате одного последовательного просмотра элементы, значение которых больше, "всплывают" образно говоря на поверхность, т.е. ближе к концу массива. Если провести такой последовательный просмотр массива несколько раз, то "тяжёлые" элементы окончательно "всплывут" и массив окажется отсортированным.

Ниже приведен пример программы. Для наглядности после каждого обмена на экран выводится текущее состояние массива. Красным цветом выделена собственно сортировка .

```
Program ex25_01;
  { ex25_01 сортировка массива методом пузырька }
  { с выводом после каждого прохода, проходы - с конца }
const
  N = 10;               { размер массива          }
Var
  i, j, buf, k: integer; { счетчики и буфер обмена }
  a: array[1..10] of integer;
begin
  { ввод массива }
    { контрольный вывод массива }
   { сортировка, проходы - с конца массива }
  writeln('сортировка:');
```

```
  for i:=1 to N do begin
    for j:=N-1 downto i do
      if a[j]>a[j+1] then begin
        buf:=a[j]; a[j]:=a[j+1]; a[j+1]:=buf;
      end;
    { вывод массива – для демонстрации }
     { итоговый вывод массива }
  end.
```

Результат работы программы:

```
исходный массив:
    7    5    1    2    9    6    4    1    3    8
сортировка:
    1    7    5    1    2    9    6    4    3    8
    1    1    7    5    2    3    9    6    4    8
    1    1    2    7    5    3    4    9    6    8
    1    1    2    3    7    5    4    6    9    8
    1    1    2    3    4    7    5    6    8    9
    1    1    2    3    4    5    7    6    8    9
    1    1    2    3    4    5    6    7    8    9
    1    1    2    3    4    5    6    7    8    9
    1    1    2    3    4    5    6    7    8    9
    1    1    2    3    4    5    6    7    8    9
отсортированный массив:
    1    1    2    3    4    5    6    7    8    9
```

Пример организации прохода с начала массива:

```
for i:=1 to N-1 do begin
  for j:=1 to N-1 do begin
    if a[j]>a[j+1] then begin
      buf:=a[j]; a[j]:=a[j+1]; a[j+1]:=buf;
     end;
   end;
  { вывод массива для отладочного контроля }
  for j:=1 to N do write(a[j]:5); writeln;
end;
```

Результат работы программы ex25_02:

```
исходный массив:
    7    5    1    2    9    6    4    1    3    8
сортировка:
    5    1    2    7    6    4    1    3    8    9
    1    2    5    6    4    1    3    7    8    9
    1    2    5    4    1    3    6    7    8    9
    1    2    4    1    3    5    6    7    8    9
    1    2    1    3    4    5    6    7    8    9
```

```
    1    1    2    3    4    5    6    7    8    9
    1    1    2    3    4    5    6    7    8    9
    1    1    2    3    4    5    6    7    8    9
    1    1    2    3    4    5    6    7    8    9
отсортированный массив:
    1    1    2    3    4    5    6    7    8    9
```

Часто оказывается, что после очередной перестановки обмен значениями уже не требуется. Чтобы в этом случае не выполнять повторение просмотров, используют вспомогательную переменную (флаг).

В приведенном выше примере сортировку можно было прекращать уже после 6-го прохода.

Начальное значение такой переменной-флага равно нулю, а в случае хотя бы одной перестановки ей присваивается ненулевое значение. Сортировка заканчивается, если перестановок не произошло (т.е. значение флага осталось нулевым). В качестве флага можно использовать логическую переменную:

```
repeat
  flag:=0;   { сбросить флаг }
  for i:=N-1 downto 1 do begin
    if a[i]>a[i+1] then begin
      buf:=a[i]; a[i]:=a[i+1]; a[i+1]:=buf;
      flag:=flag+1; f:=true;
    end;
  end;
  write (flag,' ',f:5,': ');
  for k:=1 to N do write(a[k]:5); writeln;
until flag=0 {f=false;}
```

Результат работы программы ex25_03 (в начале каждой строки приведено значение переменной **flag**:

```
8:     5    1    2    7    6    4    1    3    8    9
6:     1    2    5    6    4    1    3    7    8    9
3:     1    2    5    4    1    3    6    7    8    9
3:     1    2    4    1    3    5    6    7    8    9
2:     1    2    1    3    4    5    6    7    8    9
1:     1    1    2    3    4    5    6    7    8    9
0:     1    1    2    3    4    5    6    7    8    9
отсортированный массив:
    1    1    2    3    4    5    6    7    8    9
```

Алгоритм сортировки выбором

Очевидно, что первое место в массиве должен занять минимальный элемент массива, второе - наименьший из всех остальных, третий - наименьший из оставшихся и т.д.

Для этого необходимо выполнить следующую последовательность действий:

1. Определить минимальный элемент массива;
2. Поменять его местами с первым элементом;
3. Определить минимальный элемент среди оставшихся;
4. Поменять его местами со вторым элементом и т.д.;

Эта последовательность действий должна выполняться до тех пор, по-ка не будет определён последний минимальный элемент.

Всю операцию по упорядочиванию массива можно разбить на более простые задачи.

Вспомогательная – ввод массива. Пока ввод массива осуществим с клавиатуры (в будущем вы изучите чтение из файла):

```
{ ввод массива }
```

Для контроля введенных значений рекомендуется вывести массив на экран.

Первая задача – поиск минимального элемента. При этом важно запомнить не столько само значение минимального элемента, сколько его но-мер (индекс):

```
num:=i;
for j:=i+1 to N do
  if a[j]<a[num] then num:=j;
```

Вторая – поменять местами минимальный элемент с `i`-ым (очередным):

```
{ меняем местами a[num] и a[i] }
buf:=a[i]; a[i]:=a[num]; a[num]:=buf;
```

В приведенном ниже примере для наглядности после каждого обмена на экран также выводится текущее состояние массива.

```
Program ex25_04;
  { ex25_04 сортировка массива методом выбора }
  { с выводом массива после каждого прохода }
const
  N = 10;                { размер массива          } Var
  a: array[1..N] of integer;
  i: integer;  { счетчик                                    }
  num: integer;{ номер минимального эл-та в части от i до N }
  j: integer;  { номер эл-та, сравниваемого с минимальным   }
  buf: integer;{ буфер для обмена значениями                }
  k: integer;  { счетчик для ввода/вывода массива           }
begin
  { ввод массива }
```

```
{ сортировка }

for i:=1 to N-1 do begin
  { ищем минимальный элемент в части от i до N }
  num:=i;
  for j:=i+1 to N do
    if a[j]<a[num] then num:=j;
  { меняем местами a[num] и a[i] }
  buf:=a[i]; a[i]:=a[num]; a[num]:=buf;
  { вывод промежуточного состояния массива     }
  for k:=1 to N do write(a[k]:5); writeln;
end;
  end.
```

Результат работы програмы:

```
исходный массив:
    7    5    1    2    9    6    4    1    3    8
сортировка
    1    5    7    2    9    6    4    1    3    8
    1    1    7    2    9    6    4    5    3    8
    1    1    2    7    9    6    4    5    3    8
    1    1    2    3    9    6    4    5    7    8
    1    1    2    3    4    6    9    5    7    8
    1    1    2    3    4    5    9    6    7    8
    1    1    2    3    4    5    6    9    7    8
    1    1    2    3    4    5    6    7    9    8
    1    1    2    3    4    5    6    7    8    9
отсортированный массив:
    1    1    2    3    4    5    6    7    8    9
```

Описанные методы сортировки массивов можно применять как для сортировки массивов по возрастанию, так и для сортировки массивов по убыванию. Для этого просто необходимо определять не минимальный эле-мент массива, а максимальный. В тексте программы это выражается заменой в некоторых местах знака "<" на знак ">".