Лабораторная работа №6
Тема: Сортировка массивов

Цель работы: приобретение навыков выполнения стандартных опера-ций сортировки массивов

После выполненных работ студент должен

знать: алгоритмы сортировки массивов методом пузырька и методом выбора

уметь: создавать программы сортировки массивов

Вводный инструктаж .

Сортировка методом пузырька (обменами)

Алгоритм сортировки выбором

Вводный инструктаж

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

В обычном "неотсортированном" массиве информацию найти можно, но только путем 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.

На данном занятии будут рассмотрены два наиболее популярных ме-тода сортировки.

Массив можно создавать как путем ввода данных с клавиатуры, так и путем заполнения случайными числами. Во втором случае используются функции:

```
random;  // случайное вещ. число в интервале (0;1)
random(B); // случайное целое число в интервале [0;a)
```

Пример использования процедуры заполнения:

```
const N = 10;  // размер массива
type mass10 = array[1..N] of integer;  // объявление
типа
Var a: mass;

procedure FillMass(var m:mass;B:integer);
{ B – граница диапазона генерирования случайных чисел }
begin
  randomize;
  for i:=1 to N do m[i]:=random(B);
end;
```

В головной программе указывается вызов процедуры:

```
FillMass(a,P); // P – граница диапазона, напр. 100
```

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 описан-ным выше способом. Так в результате одного последовательного просмотра элементы, значение которых больше, "всплывают" образно говоря на поверх-ность, т.е. ближе к концу массива. Если провести такой последовательный просмотр массива несколько раз, то "легкие" элементы окончательно "всплы-вут" и массив окажется отсортированным.

Ниже приведен пример программы.

```
Program ex26_01;
  { ex26_01 сортировка массива методом пузырька    }
  { проходы с конца массива
} 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 на экран }
   { сортировка }
  for i:=1 to N do begin
    for j:=N-1 downto i do
      { сравниваем соседние элементы массива}
      if a[j]>a[j+1] then begin
        { если стоят "неправильно" - меняем местами }
        buf:=a[j]; a[j]:=a[j+1]; a[j+1]:=buf;
      end;
  end;
  { итоговый вывод массива }
end.
```

Пример организации прохода с начала массива (показан только фрагмент программы, содержащий сортировку):

```
{ сортировка }
for i:=1 to N-1 do begin
  for k:=1 to N-1 do begin
    { сравниваем соседние элементы массива}
    if a[k]>a[k+1] then begin
     { если стоят "неправильно" - меням местами }
      buf:=a[k]; a[k]:=a[k+1]; a[k+1]:=buf;
    end;
  end;
end;
```

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

```
{ сортировка }
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;
until flag=0 {f=false;}
```

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**:

```
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;
```

Пример программы:

```
Program ex25_02;
{ ex25_02 сортировка массива методом выбора }
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; { индекс минимального элемента }
  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;
end;
 end.
```

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 ">".

Задания для самостоятельного выполнения

Во всех заданиях необходимо вывести на экран исходный массив и массив после сортировки.

Для получения оценки «отлично» создать процедуру заполнения массива случайными числами.

1. Организуйте массив, содержащий 10 различных целых чисел, после этого 5 первых элементов массива упорядочиваются по возрастанию, а 5 по-следних элементов по убыванию. Содержимое отсортированного таким об-разом массива выводится на экран. *Указание. В программе записать два бло-ка сортировки: один для первой части массива, другой для второй части.*

2. Организуйте массив, содержащий 15 различных целых чисел после этого отдельно первых 5 элементов, вторых 5 элементов и последних 5 эле-ментов сортируются по возрастанию. Содержимое отсортированного таким образом массива выводится на экран. *Указание. В программе записать три блока сортировки: отдельно для каждой части массива.*

3. Создайте массив содержаний 10 чисел типа **`real`**. Отсортируйте его по убыванию, после этого определите и выведите на экран сумму элементов с чётными индексами и сумму элементов с нечётными индексами.

4. Организуйте массив, содержащий 10 различных символов. Отсорти-руйте его по возрастанию. *Указание. Для символов применимы операции сравнения аналогично целым числам. Не забудьте о совместимости типов. Для генерации случайных символов используйте функцию* **`a[i]:=chr(random(26)+65)`**

5. Организуйте массив содержащий 10 символов. Отсортируйте отдельно 5 первых элементов по возрастанию, а 5 последних элементов по убыванию. *Указание. Для символов применимы операции сравнения аналогично целым числам.*

6. Создайте массив, содержаний 10 различных целых чисел. Содержимое массива сортируется по убыванию, и после этого определяются минимальный и максимальный элементы массива.

7. Создайте массив, содержащий 15 чисел типа **`real`**. Отдельно первых 5 элементов массива, вторых 5 элементов и последних 5 элементов отсортируйте по убыванию. *Указание. В программе записать три блока сортировки: отдельно для каждой части массива.*

8. Создайте массив содержащий 25 чисел типа **`real`.** Отдельно первых 5 элементов массива, вторых 5 элементов и последующих по 5 элементов отсортируйте по возрастанию. *Указание. В программе записать блоки сортировки: отдельно для каждой части массива.*

9. Создайте массив, содержащий 10 различных символов. Отсортируйте его по убыванию. После этого определите и выведите на экран "наименьший" и "наибольший" символы.

10. Создайте массив, содержащий 10 различных символов. Первую и вторую половину массива отсортируйте по возрастанию. Отсортированный массив выведите на экран. *Указание. Для символов применимы операции сравнения аналогично целым числам.*

11. Создайте целочисленный массив A, содержащий 10 различных чисел. Отсортируйте первые 8 элементов по убыванию.

12. Создайте массив, содержащий 10 различных целых чисел. Отсортируйте его по убыванию. После этого замените все элементы массива на противоположные и выведите содержимое обработанного массива на экран

.