ФЕДЕРАЛЬНОЕ АГЕНТСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ДИМИТРОВГРАДСКИЙ ИНТСТИТУТ ТЕХНОЛОГИЙ, УПРАВЛЕНИЯ И ДИЗАЙНА
“КАФЕДРА ВЫСШЕЙ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ”

Лабораторная работа №6
по курсу “Алгоритмы и Структуры данных”
на тему: “Сортировки данных в памяти”
Вариант № 19

Выполнил студент группы ВТ-21:
Потеренко А.Г.
Проверил преподаватель:
Мингалиев Р.Ш.

Димитровград 2005г.

Порядок работы.

1. Анализ индивидуального задания и разработка способов представления объектов задачи в памяти, методов доступа к ним.
2. Разработка программы на языке Паскаль.
3. Разработка контрольных примеров.
4. Отладка программ.
5. Составление отчета.

Содержание отчета.

Общие требования.

Лабораторные работы могут быть выполнены на любом языке программирования. Задачи должны быть реализованы в общем случае полностью. Обязательно за собой восстанавливать полностью исходное состояние ЭВМ (очистка ОП, восстановление режима экрана и т.п.). Лабораторные работы должны быть снабжены комментариями, и содержать описание задачи и представление автора. Программа должна адекватно реагировать на любые действия пользователя и контролировать вводимую информацию на допустимость значений. Наличие интерфейса в виде меню с возможностью добавления, удаления, редактирования данных обязательно.

1. Текст постановки задачи.

Написать программу, реализующую сортировки:

- **Метод пузырька.**
- **Нерекурсивная версия Quicksort.**

1. Режим демонстрации алгоритма сортировки (число элементов 2 - 20, задаются случайным образом или вводятся с клавиатуры)
2. Сортировка массива из 32, 256, 4096, 16384 элементов (массив полностью упорядочен, упорядочен в обратном порядке, задан случайным образом). Определить время выполнения процедуры сортировки.
3. Сортировка массива из 32, 256, 4096, 16384 элементов (массив полностью упорядочен, упорядочен в обратном порядке, задан случайным образом). Определить число сравнений и перестановок элементов.

2. Изложение способов представления объектов задачи в памяти и методов доступа к ним.

В данной задаче элементы в памяти представляются статическим массивом. Таким образом, память выделяется на этапе компиляции программы, а не выделяется динамически. Соответственно, доступ к объектам задачи осуществляется как к элементам массива.

3. Алгоритм решения данной задачи.

1. Метод пузырька.

Простая обменная сортировка (в просторечии называемая "методом пузырька") для массива a[1], a[2], ..., a[n] работает следующим образом. Начиная с конца массива сравниваются два соседних элемента (a[n] и a[n-1]). Если выполняется условие a[n-1] > a[n], то значения элементов меняются местами. Процесс продолжается для a[n-1] и a[n-2] и т.д., пока не будет произведено сравнение a[2] и a[1]. Понятно, что после этого на месте a[1] окажется элемент массива с наименьшим значением. На втором шаге процесс повторяется, но последними сравниваются a[3] и a[2]. И так далее. На последнем шаге будут сравниваться только текущие значения a[n] и a[n-1]. Понятна аналогия с пузырьком, поскольку наименьшие элементы (самые "легкие") постепенно "всплывают" к верхней границе массива. Пример сортировки методом пузырька показан в таблице 2.3.

Таблица 2.3. Пример сортировки методом пузырька

Начальное состояние массива	8 23 5 65 44 33 1 6
Шаг 1	8 23 5 65 44 33 1 6 8 23 5 65 44 1 33 6 8 23 5 65 1 44 33 6 8 23 5 1 65 44 33 6 8 23 1 5 65 44 33 6 8 1 23 5 65 44 33 6 1 8 23 5 65 44 33 6
Шаг 2	1 8 23 5 65 44 6 33 1 8 23 5 65 6 44 33 1 8 23 5 6 65 44 33 1 8 23 5 6 65 44 33 1 8 5 23 6 65 44 33 1 5 8 23 6 65 44 33
Шаг 3	1 5 8 23 6 65 33 44 1 5 8 23 6 33 65 44 1 5 8 23 6 33 65 44 1 5 8 6 23 33 65 44 1 5 6 8 23 33 65 44
Шаг 4	1 5 6 8 23 33 44 65 1 5 6 8 23 33 44 65 1 5 6 8 23 33 44 65 1 5 6 8 23 33 44 65
Шаг 5	1 5 6 8 23 33 44 65 1 5 6 8 23 33 44 65 1 5 6 8 23 33 44 65
Шаг 6	1 5 6 8 23 33 44 65 1 5 6 8 23 33 44 65
Шаг 7	1 5 6 8 23 33 44 65

Для метода простой обменной сортировки требуется число сравнений $n^x(n-1)/2$, минимальное число пересылок 0, а среднее и максимальное число пересылок - O(n2).

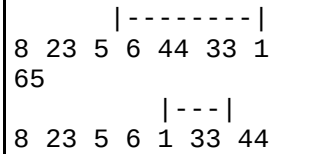
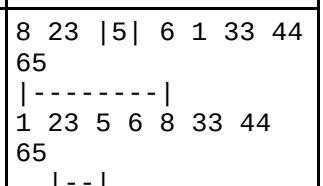
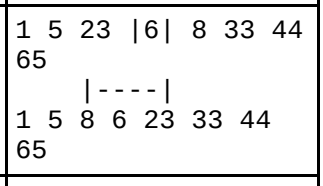
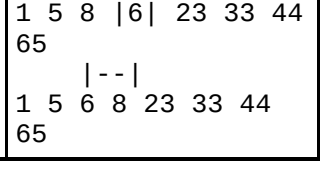
Метод пузырька допускает три простых усовершенствования. Во-первых, как показывает таблица 2.3, на четырех последних шагах расположение значений элементов не менялось (массив оказался уже упорядоченным). Поэтому, если на некотором шаге не было произведено ни одного обмена, то выполнение алгоритма можно прекращать. Во-вторых, можно запоминать наименьшее значение индекса массива, для которого на текущем шаге выполнялись перестановки. Очевидно, что верхняя часть массива до элемента с этим индексом уже отсортирована, и на следующем шаге можно прекращать сравнения значений соседних элементов при достижении такого значения индекса. В-третьих, метод пузырька работает неравноправно для "легких" и "тяжелых" значений. Легкое значение попадает на нужное место за один шаг, а тяжелое на каждом шаге опускается по направлению к нужному месту на одну позицию.

2. Нерекурсивная версия Quicksort.

Метод сортировки разделением был предложен Чарльзом Хоаром (он любит называть себя Тони) в 1962 г. Этот метод является развитием метода простого обмена и настолько эффективен, что его стали называть "методом быстрой сортировки - Quicksort".

Основная идея алгоритма состоит в том, что случайным образом выбирается некоторый элемент массива x, после чего массив просматривается слева, пока не встретится элемент a[i] такой, что a[i] > x, а затем массив просматривается справа, пока не встретится элемент a[j] такой, что a[j] < x. Эти два элемента меняются местами, и процесс просмотра, сравнения и обмена продолжается, пока мы не дойдем до элемента x. В результате массив окажется разбитым на две части - левую, в которой значения ключей будут меньше x, и правую со значениями ключей, большими x. Далее процесс рекурсивно продолжается для левой и правой частей массива до тех пор, пока каждая часть не будет содержать в точности один элемент. Понятно, что как обычно, рекурсию можно заменить итерациями, если запоминать соответствующие индексы массива. Проследим этот процесс на примере нашего стандартного массива (таблица 2.6).

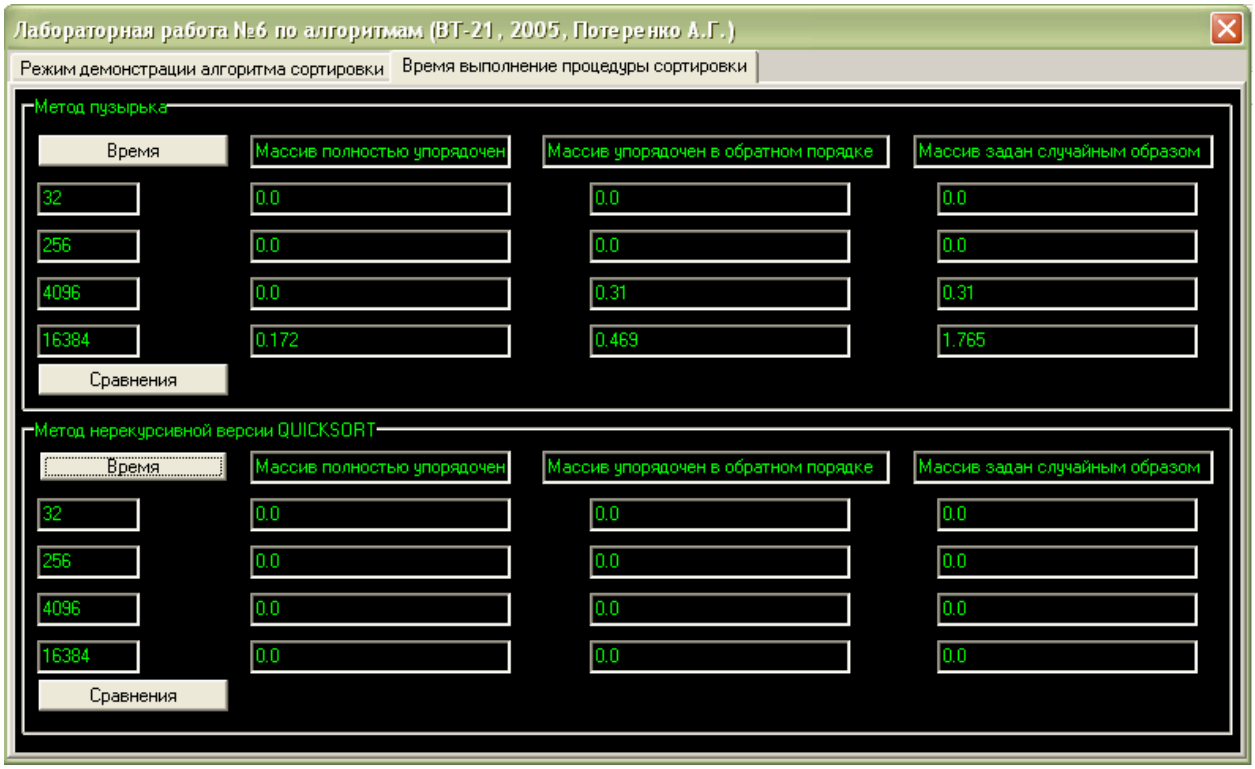
Таблица 2.6. Пример быстрой сортировки

Начальное состояние массива	8 23 5 65 |44| 33 1 6
Шаг 1 (в качестве x выбирается a[5])	
Шаг 2 (в подмассиве a[1], a[5] в качестве x выбирается a[3])	
Шаг 3 (в подмассиве a[3], a[5] в качестве x выбирается a[4])	
Шаг 4 (в подмассиве a[3], a[4] выбирается a[4])	

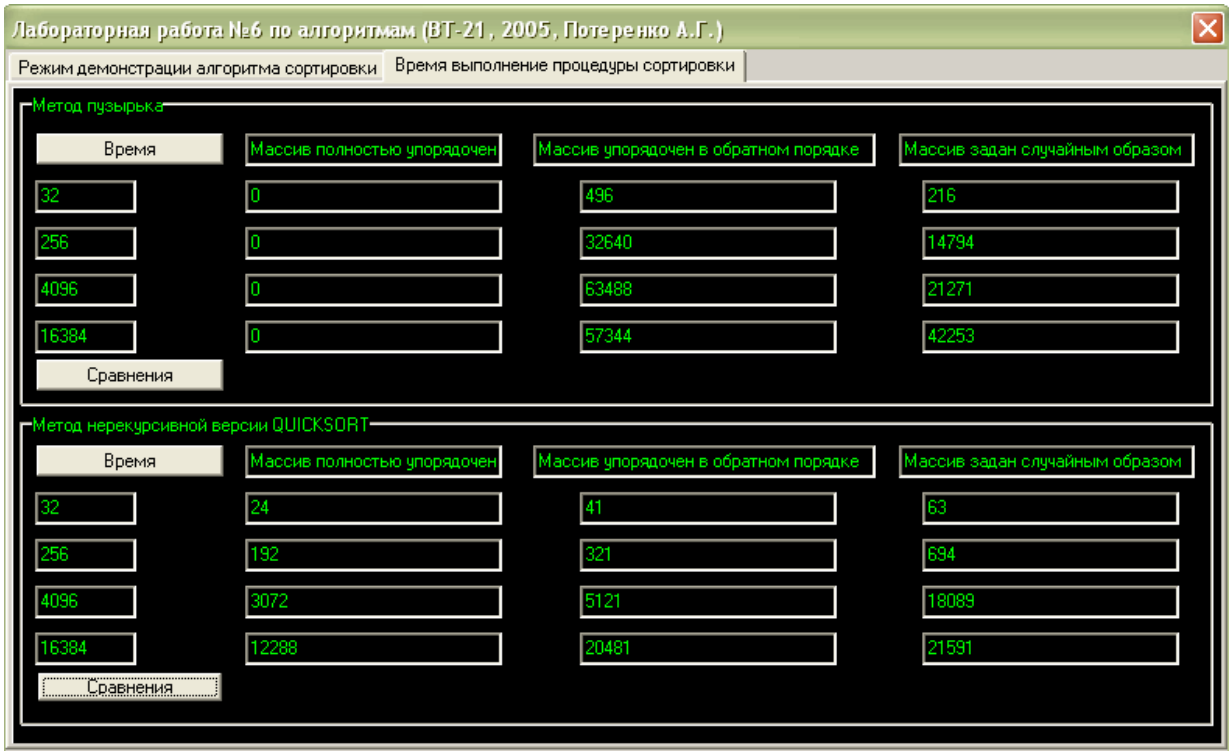
Алгоритм недаром называется быстрой сортировкой, поскольку для него оценкой числа сравнений и обменов является O(n*log n). На самом деле, в большинстве утилит, выполняющих сортировку массивов, используется именно этот алгоритм.

4. Описание и обоснование контрольных примеров.

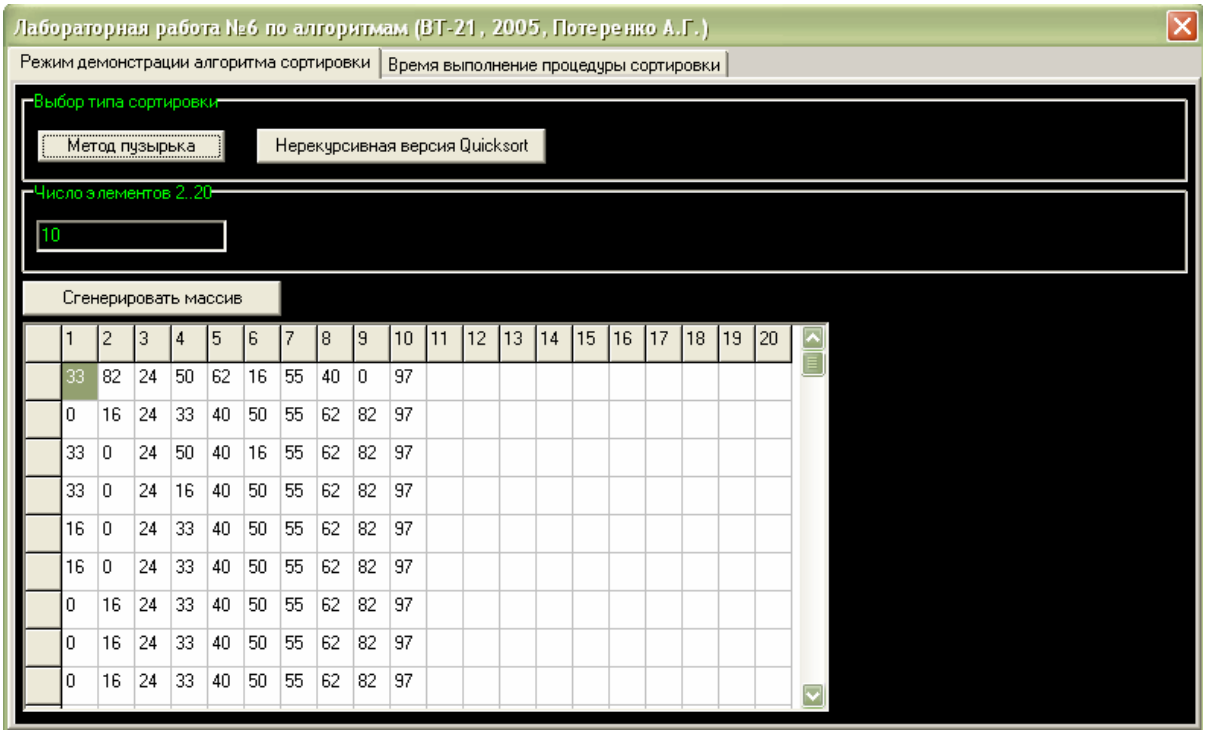
Тестирование программы осуществляется с помощью проверки ее на работоспособность при различных входных параметрах. В данном случае осуществляется защита от неправильных входных данных. Как видно на изображениях ниже, программа адекватно реагирует на события, производимые пользователем.
На данном рисунке показаны расчеты программы – время выполнения сортировки массива при четырех значениях количества n элементов. Можно заметить, что время второй сортировки невозможно измерить при таком количестве элементов.



На данном рисунке показаны расчеты программы – количество перестановок элементов массива при четырех значениях количества n элементов.



Результаты работы программы при различных n – от 2 до 20 элементов.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	33	82	24	50	62	16	55	40	0	97										
	0	16	24	33	40	50	55	62	82	97										
	33	0	24	50	40	16	55	62	82	97										
	33	0	24	16	40	50	55	62	82	97										
	16	0	24	33	40	50	55	62	82	97										
	16	0	24	33	40	50	55	62	82	97										
	0	16	24	33	40	50	55	62	82	97										
	0	16	24	33	40	50	55	62	82	97										
	0	16	24	33	40	50	55	62	82	97										

5. Текст программы с комментариями (в виде приложения).

Исходный текст программы:

```
program Project1;

uses
  Forms,
  Unit1 in 'Unit1.pas' {Form1};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.Run;
end.
```

Исходный текст главного модуля программы:

```
unit Unit1;

……

const
  n=16384;
var
  Form1: TForm1;
  i: word;
  mil,sec,delta,deltas: longint;
  Mas:array[1..n] of integer;
  //-----------------------------------------------------------------------------
  NewPage: TTabSheet;
  NewPanel: TPanel;
  NewButton: TButton;
  NewGroupBox: TGroupBox;
  NewEdit1, NewEdit2, NewEdit3, NewEdit4, NewEdit5, NewEdit6: TEdit;
  NewEdit7, NewEdit8, NewEdit9, NewEdit10: TEdit;
  NewEdit11, NewEdit12, NewEdit13, NewEdit14: TEdit;
  NewEdit15, NewEdit16, NewEdit17, NewEdit18: TEdit;
  NewEdit19, NewEdit20, NewEdit21, NewEdit22: TEdit;
  NewEdit23, NewEdit24, NewEdit25, NewEdit26: TEdit;
  NSG: TStringGrid;
  NL: TLabel;
  GLT:TSystemTime;

……

procedure TForm1.BEGIN3(Sender: TObject);
var i,j:word;
begin
  for i:=1 to n do NSG.Cells[i,1]:='';
  Randomize;
  try
  //-----------------------------------------------------------------------------
  if (StrToInt(NewEdit1.Text)>0)and(StrToInt(NewEdit1.Text)<21) then
  for i:=1 to StrToInt(NewEdit1.Text) do
   begin
     Mas[i]:=random(100);
     NSG.Cells[i,1]:=VarToStr(Mas[i]);
   end;
  for j:=2 to NSG.RowCount do
    for i:=1 to 20 do NSG.Cells[i,j]:='';
  NSG.RowCount:=2;
  //-----------------------------------------------------------------------------
  except
  end;
end;
//-----------------------------------------------------------------------------
procedure TForm1.BEGIN4_VR(Sender: TObject);            //Время методом пузырька
const masser:array[1..4] of word=(32,256,4096,16384);
var x,i,j,p,tre:word;
begin
```

```
  randomize;
  for tre:=1 to 3 do
  for p:=1 to 4 do
  begin
  case tre of
    1:for i:=1 to  masser[p] do Mas[i]:=i;
    2:for i:=1 to  masser[p] do Mas[i]:=masser[p]-i;
    3:for i:=1 to  masser[p] do Mas[i]:=random(100);
   end;
  //---------------------------------------------------------------------------
  GetLocalTime(GLT);
  mil:=GLT.wMilliseconds;
  sec:=GLT.wSecond;
  for i:=2 to masser[p] do
   for j:=masser[p] downto i do
    if Mas[j-1]>Mas[j] then
     begin
       x:=Mas[j-1];
       Mas[j-1]:=Mas[j];
       Mas[j]:=x;
     end;
  //---------------------------------------------------------------------------
  GetLocalTime(GLT);
  if (GLT.wSecond-sec)>=0 then deltas:=GLT.wSecond-sec
             else
               begin
                deltas:=60-sec;
                deltas:=deltas+GLT.wSecond;
               end;
  if (GLT.wMilliseconds-mil)>=0 then delta:=GLT.wMilliseconds-mil
             else
               begin
                delta:=1000-mil;
                delta:=delta+GLT.wMilliseconds;
               end;
  case tre of
     1:begin
            case p of
               1:NewEdit3.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               2:NewEdit4.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               3:NewEdit5.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               4:NewEdit6.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
            end;
       end;
     2:begin
           case p of
               1:NewEdit7.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               2:NewEdit8.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               3:NewEdit9.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               4:NewEdit10.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
            end;
       end;
     3:begin
          case p of
               1:NewEdit11.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               2:NewEdit12.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               3:NewEdit13.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               4:NewEdit14.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
            end;
       end;
    end;
  end;
end;
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
//---------------------------------------------------------------------------
procedure TForm1.BEGIN5_VR(Sender: TObject);           //Время методом QUISKSORT
const masser:array[1..4] of longint=(32,256,4096,16384);
type gop=record
            L:longint;
            R:longint;
           end;
var x,i,j,p,tre:longint;
    stack:array[1..n] of gop;
    s,n,L,R,w:longint;
```

```
begin
  randomize;
  for tre:=1 to 3 do
  for p:=1 to 4 do
  begin
  case tre of
    1:for i:=1 to  masser[p] do Mas[i]:=i;
    2:for i:=1 to  masser[p] do Mas[i]:=masser[p]-i;
    3:for i:=1 to  masser[p] do Mas[i]:=random(100);
   end;
  //-------------------------------------------------------------------------
  GetLocalTime(GLT);
  mil:=GLT.wMilliseconds;
  sec:=GLT.wSecond;
  //-------------------------------------------------------------------------
  n:=masser[p];
  s:=1;
  stack[1].L:=1;
  stack[s].R:=n;
  repeat
    L:=stack[s].L;
    R:=stack[s].R;
    s:=s-1;
    repeat
      i:=L;
      j:=R;
      x:=Mas[(L+R) div 2];
      repeat
         while Mas[i]<x do begin i:=i+1 end;
         while x<Mas[j] do begin j:=j-1; end;
         if i<=j then
           begin
             w:=Mas[i];
             Mas[i]:=Mas[j];
             Mas[j]:=w;
             i:=i+1;
             j:=j-1;
           end;
      until i>j;
      if i<R then
        begin
          s:=s+1;
          stack[s].L:=i;
          stack[s].R:=R;
        end;
      R:=j;
    until L>=R;
  until s=0;
  //-------------------------------------------------------------------------
  GetLocalTime(GLT);
  if (GLT.wSecond-sec)>=0 then deltas:=GLT.wSecond-sec
             else
               begin
                deltas:=60-sec;
                deltas:=deltas+GLT.wSecond;
               end;
  if (GLT.wMilliseconds-mil)>=0 then delta:=GLT.wMilliseconds-mil
             else
               begin
                delta:=1000-mil;
                delta:=delta+GLT.wMilliseconds;
               end;
  case tre of
     1:begin
            case p of
               1:NewEdit15.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               2:NewEdit16.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               3:NewEdit17.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               4:NewEdit18.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
            end;
       end;
     2:begin
           case p of
               1:NewEdit19.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               2:NewEdit20.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
```

```
               3:NewEdit21.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               4:NewEdit22.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
             end;
       end;
     3:begin
          case p of
               1:NewEdit23.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               2:NewEdit24.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               3:NewEdit25.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
               4:NewEdit26.Text:=VarToStr(deltas)+'.'+VarToStr(delta);
             end;
       end;
    end;
  end;
end;
//---------------------------------------------------------------------------
procedure TForm1.BEGIN6_PER(Sender: TObject);//Число перестановок методом пузырька
const masser:array[1..4] of word=(32,256,4096,16384);
var x,i,j,p,tre:word;
    CON1:word;
begin
  for tre:=1 to 3 do
  for p:=1 to 4 do
  begin
  randomize;
  case tre of
    1:for i:=1 to  masser[p] do Mas[i]:=i;
    2:for i:=1 to  masser[p] do Mas[i]:=masser[p]-i;
    3:for i:=1 to  masser[p] do Mas[i]:=random(100);
   end;
  //---------------------------------------------------------------------------
  CON1:=0;
  for i:=2 to masser[p] do
   for j:=masser[p] downto i do
    if Mas[j-1]>Mas[j] then
     begin
       INC(CON1);
       x:=Mas[j-1];
       Mas[j-1]:=Mas[j];
       Mas[j]:=x;
     end;
  //---------------------------------------------------------------------------
  case tre of
     1:begin
            case p of
               1:NewEdit3.Text:=VarToStr(CON1);
               2:NewEdit4.Text:=VarToStr(CON1);
               3:NewEdit5.Text:=VarToStr(CON1);
               4:NewEdit6.Text:=VarToStr(CON1);
             end;
       end;
     2:begin
           case p of
               1:NewEdit7.Text:=VarToStr(CON1);
               2:NewEdit8.Text:=VarToStr(CON1);
               3:NewEdit9.Text:=VarToStr(CON1);
               4:NewEdit10.Text:=VarToStr(CON1);
             end;
       end;
     3:begin
          case p of
               1:NewEdit11.Text:=VarToStr(CON1);
               2:NewEdit12.Text:=VarToStr(CON1);
               3:NewEdit13.Text:=VarToStr(CON1);
               4:NewEdit14.Text:=VarToStr(CON1);
             end;
       end;
    end;
  end;
end;
//---------------------------------------------------------------------------
procedure TForm1.BEGIN7_PER(Sender: TObject);//Число перестановок методом QUISKSORT
const masser:array[1..4] of longint=(32,256,4096,16384);
type gop=record
            L:longint;
```

```
            R:longint;
           end;
var x,i,j,p,tre:longint;
    stack:array[1..n] of gop;
    s,n,L,R,w:longint;
    CON1:word;
begin
  randomize;
  for tre:=1 to 3 do
  for p:=1 to 4 do
  begin
  case tre of
    1:for i:=1 to  masser[p] do Mas[i]:=i;
    2:for i:=1 to  masser[p] do Mas[i]:=masser[p]-i;
    3:for i:=1 to  masser[p] do Mas[i]:=random(100);
   end;
  //-------------------------------------------------------------------------
  CON1:=0;
  n:=masser[p];
  s:=1;
  stack[1].L:=1;
  stack[s].R:=n;
  repeat
    L:=stack[s].L;
    R:=stack[s].R;
    s:=s-1;
    repeat
      i:=L;
      j:=R;
      x:=Mas[(L+R) div 2];
      repeat
         while Mas[i]<x do begin i:=i+1 end;
         while x<Mas[j] do begin j:=j-1; end;
         if i<=j then
           begin
             INC(CON1);
             w:=Mas[i];
             Mas[i]:=Mas[j];
             Mas[j]:=w;
             i:=i+1;
             j:=j-1;
           end;
      until i>j;
      if i<R then
        begin
          INC(CON1);
          s:=s+1;
          stack[s].L:=i;
          stack[s].R:=R;
        end;
      R:=j;
    until L>=R;
  until s=0;
  //-------------------------------------------------------------------------
  case tre of
     1:begin
            case p of
               1:NewEdit15.Text:=VarToStr(CON1);
               2:NewEdit16.Text:=VarToStr(CON1);
               3:NewEdit17.Text:=VarToStr(CON1);
               4:NewEdit18.Text:=VarToStr(CON1);
             end;
       end;
     2:begin
           case p of
               1:NewEdit19.Text:=VarToStr(CON1);
               2:NewEdit20.Text:=VarToStr(CON1);
               3:NewEdit21.Text:=VarToStr(CON1);
               4:NewEdit22.Text:=VarToStr(CON1);
             end;
       end;
     3:begin
          case p of
               1:NewEdit23.Text:=VarToStr(CON1);
               2:NewEdit24.Text:=VarToStr(CON1);
```

```
              3:NewEdit25.Text:=VarToStr(CON1);
              4:NewEdit26.Text:=VarToStr(CON1);
            end;
       end;
    end;
  end;
end;
//---------------------------------------------------------------------------
procedure TForm1.BEGIN1(Sender: TObject);                    //Метод пузырька
var x,i,j,n,p,e:word;
begin
 try
  p:=2;
  n:=StrToInt(NewEdit1.Text);
  NSG.RowCount:=2;
  for i:=2 to n do
   for j:=n downto i do
    if Mas[j-1]>Mas[j] then
     begin
       x:=Mas[j-1];
       Mas[j-1]:=Mas[j];
       Mas[j]:=x;
       //---------------------------------------------------------------------
       for e:=1 to StrToInt(NewEdit1.Text) do NSG.Cells[e,p]:=VarToStr(Mas[e]);
       INC(P);
       NSG.RowCount:=NSG.RowCount+1
       //---------------------------------------------------------------------
     end;
 except
 end;
end;
//---------------------------------------------------------------------------
procedure TForm1.BEGIN2(Sender: TObject);       //Нерекурсивная версия QUICKSORT
type gop=record
           L:word;
           R:word;
          end;
var stack:array[1..n] of gop;
    s,n,L,R,j,x,w,e,p:word;
begin
 try
  n:=StrToInt(NewEdit1.Text);
  s:=1;
  stack[1].L:=1;
  stack[s].R:=n;
  NSG.RowCount:=2;
  p:=2;
  repeat
    L:=stack[s].L;
    R:=stack[s].R;
    s:=s-1;
    repeat
      i:=L;
      j:=R;
      x:=Mas[(L+R) div 2];
      repeat
         while Mas[i]<x do begin i:=i+1 end;
         while x<Mas[j] do begin j:=j-1; end;
         if i<=j then
           begin
             w:=Mas[i];
             Mas[i]:=Mas[j];
             Mas[j]:=w;
             i:=i+1;
             j:=j-1;
             //---------------------------------------------------------------
             for e:=1 to StrToInt(NewEdit1.Text) do NSG.Cells[e,p]:=VarToStr(Mas[e]);
             INC(P);
             NSG.RowCount:=NSG.RowCount+1
             //---------------------------------------------------------------
           end;
      until i>j;
      if i<R then
        begin
          s:=s+1;
```

```
          stack[s].L:=i;
          stack[s].R:=R;
        end;
      R:=j;
    until L>=R;
  until s=0;
  //---------------------------------------------------------------------------
  for i:=1 to StrToInt(NewEdit1.Text) do NSG.Cells[i,2]:=VarToStr(Mas[i]);
 except
 end;
end;
//---------------------------------------------------------------------------
end.
```