ALGORITMI ZA SORTIRANJE TABEL z napotki za izdelavo
1. Seminarske naloge

```
=========================================================================
const n = …;
type item = record
              key: integer;
            end;
     index = 0..n;
var a: array [0..n] of item;

(*   Upostevajte, da v seminarski nalogi dejansko urejamo trenutne  *)
(*   tabele manjse od n, torej je n najvecja velikost tabele        *)

=========================================================================
procedure StraightInsertion;  (* navadno vstavljanje *)
var i,j: index;
    x: item;
begin
  for i := 2 to n do
  begin
    x := a[i];
    a[0] := x;
    j := i - 1;
    while x.key < a[j].key do
    begin
      a[j+1] := a[j];
      j := j - 1;
    end;
    a[j+1] := x
  end
end;

=========================================================================
procedure BinaryInsertion;  (* dvojisko vstavljanje *)
var i, j, l, r, m: index;
    x: item;
begin
  for i := 2 to n do
  begin
    x := a[i];
    l := 1;
    r := i - 1;
    while l <= r do
    begin
      m := (l+r) div 2;
      if x.key < a[m].key then
        r := m - 1
      else
        l := m + 1;
    end;
    for j := i - 1 downto l do
      a[j+1] := a[j];
    a[l]:=x;
  end
end;
```

```
=========================================================================
procedure StraightSelection;   (* navadno izbiranje *)
var i,j,k: index;
      x: item;
begin
  for i : = 1 to n - 1 do
  begin
    k : = i;
    x : = a[i];
    for j : = i + 1 to n do
      if a[j].key < x.key then
      begin
        k : = j;
        x : = a[j]
      end;
    a[k] : = a[i];
    a[i] : = x;
  end
end;

=========================================================================
procedure BubbleSort;   (* navadna zamenjava *)
var i,j: index;
    x: item;
begin
  for i := 2 to n do
    for j := n downto i do
      if a[j-1].key > a[j].key then
      begin
        x := a[j-1];
        a[j-1] := a[j];
        a[j] := x
      end
end;

=========================================================================
procedure BubbleSort2;     (* izboljsana navadna zamenjava *)

(*   Pregledovanje tabele samo do mesta zadnje zamenjave,  *)
(*   ne do konca (dodaten indeks za oznako mesta zadnje    *)
(*   zamenjave)                                            *)
(*   NAPISITE SAMI !                                       *)

=========================================================================
procedure ShakerSort;   (* sortiranje s stresanjem *)
var j,k,l,r: index;
    x: item;
begin
  l := 2;
  r := n;
  k := n;
  repeat
    for j := r downto l do
      if a[j-1].key > a[j].key then
      begin
        x := a[j-1];
```

```
        a[j-1] := a[j];
        a[j] := x;
        k := j
      end
    l := k + 1;
    for j := l to r do
      if a[j-1].key > a[j].key then
      begin
        x := a[j-1];
        a[j-1] := a[j];
        a[j] := x;
        k := j
      end
    r := k - 1
  until l > r
end;

=========================================================================
(* Tabela h ne sme biti fiksna (kot v spodnjem algoritmu), temvec     *)
(* se doloca v odvisnosti od velikosti tabele na naslednji nacin:     *)
(* t:= |log2 n|-1,   h[t]:=1,   h[k-1] := 2*h[k]+1      ALI           *)
(* t:= |log3 n|-1,   h[t]:=1,   h[k-1] := 3*h[k]+1                    *)
(* Pozor tudi na negativne indekse …                                  *)

procedure ShellSort;(*sortiranje s vstavljanjem s padajocim prirastkom*)
const  t = 4;
var i,j,k,s: index;
    x : item;
    m: 1..t;
    h: array [1..t] of integer;
begin
  h[i] := 9;
  h[2] := 5;
  h[3] := 3;
  h[4] := 1;
  for m := 1 to t do
  begin
    k := h[m];
    s := -k;
    for i := k+1 to n do
    begin
      x := a[i];
      j := i - k;
      if s = 0 then
        s := -k;
      s := s + 1;
      a[s] := x;
      while x.key < a[j].key do
      begin
        a[j+k] := a[j];
        j := j - k;
      end;
      a[j+k] := x
    end
  end
end;
```

```
=========================================================================
procedure HeapSort;  (* sortiranje s kopico - rekurzivno *)
var l,r: index;

  procedure Sift (l,r:index);
  var i,j: index;
      x: item;
      s: boolean;
  begin
    i := l;
    j := 2*i;
    x := a[i];
    s := true;  (* namesto label 13 *)
    while j <= r and s do
    begin
      if j < r then
        if a[j].key < a[j+1].key then
          j := j + 1;
      if x.key >= a[j].key then
        s := false
      else
      begin
        a[i] := a[j];
        i := j;
        j := 2 * i;
      end
    end
    a[i] := x
  end;

begin
  l := (n div 2) + 1;
  r := n;
  while l > 1 do
  begin
    l := l-1;
    Sift(l,n);
  end;
  while r > 1 do
  begin
    x := a[1];
    a[1] := a[r];
    a[r] := x;
    r := r-1;
    Sift(1,r)
  end
end;

=========================================================================
(* Ceprav seminarska naloga tega ne zahteva, razmislite o izboljsavah *)
(* spodnjega algoritma - recimo boljse izbiranje srednjega elementa … *)

procedure QuickSort;  (* sortiranje s premenami - rekurzivno *)

  procedure Sort (l,r: index);
  var i,j: index;
      x,w: item;
```

```
  begin
    i := l;
    j := r;
    x := a[(l+r) div 2];
    repeat
      while a[i].key < x.key do
        i := i + 1;
      while x.key < a[j].key do
        j := j - 1;
      if i <= j then
      begin
        w := a[i];
        a[i] := a[j];
        a[j] := w;
        i := i + 1;
        j := j - 1;
      end;
    until i > j;
    if l < j then
      sort (l,j);
    if i < r then
      sort (i,r);
  end;

begin
  sort(1,n);
end;
```

==========

```
(* In se ena od razlicic podprogramov za merjenje casa z uporabo …  *)
…
uses Dos;
…
var u,m,s,st:word;
…
procedure Zacni_Uro;
begin
  gettime(u,m,s,st);
end;

function Ustavi_Uro:Real;
var u1,m1,s1,st1:word;
begin
  gettime(u1,m1,s1,st1);
  Ustavi_Uro:=((((u1-u)*60 + m1-m)*60 + s1-s)*100+st1-st)/100;
end;
…
procedure Sort(…);
var …
begin
  …
  Zacni_Uro;
  for i:= …  do
  begin
    …
  end;
  writeln('Cas sortiranja v tisocinkah je ',round(Ustavi_Uro*1000));
end;
```