

FÖRORD

Detta kompendium innehåller lösningsförslag till de flesta av programmeringsuppgifterna i Njord/Sandmarks **Turbovägen till Pascal**.

Vid sammanställningen har vi strävat efter att ge pascalkoden en enhetlig utformning (se nedan). Vissa stilmässiga och andra inkonsekvenser kvarstår dock.

Programnotation

De regler som tillämpats sammanfattas här:

1. Två blanktecken används vid *indentering* (indragning).
2. *Reserverade ord* skrivs med fetstilta små bokstäver, t ex **begin**, **function** och **unit**.
3. Namn på enheter, datatyper, procedurer och funktioner inleds med en stor bokstav. I övrigt används små bokstäver, utom vid avgränsning av sammansatta ord. Exempel: CRT (förkortning av Cathode Ray Tube), Dagtyp, Sortera, MaxAvTre.
4. Konstanter och variabler inleds med liten bokstav, t ex `heltal` och de fördeklarerade konstanterna `false` och `true`. Stora bokstäver tillgrips även för att underlätta läsning av sammansatta namn, t ex `antalTecken`, `sorteradLista` och de fördeklarerade konstanterna `maxInt` och `maxLongInt`.
5. I **if..then..else-**, **for..do-**, **while..do-** och **with..do-** satser där satsparentesen **begin..end** används, placeras **begin** på samma rad som **then**, **else**, **do** osv, medan **end** får tydliggöra satsens slut genom att placeras på egen rad (se exemplet nedan). I förtydligande syfte upprepas ibland satstypen i en kommentar efter **end**, t ex:

```
if a < b then begin                if n >= 0 then begin
  ..                               ..
  ..                               end
end; (* if *)                       else begin
                                   ..
                                   end; (* if *)
```

Anmärkning: I vissa fall, främst för att spara utrymme, har avvikelser från notationsreglerna gjorts.

Författarna

Martin Fahlgren
Stavstigen 12A
417 02 Göteborg

Bernt Svensson
Fridkullagatan 19C
412 62 Göteborg

KAPITEL 2

```
program Ovn_2_1;  
begin  
  WriteLn('Pascal är ett exempel på ett högnivåspråk.');
```

WriteLn('Högnivåspråk kallas även för problemorienterat språk.');

```
end.
```

```
program Ovn_2_2;  
uses CRT;  
begin  
  ClrScr;  
  WriteLn('Modesty Blaise');
```

```
end.
```

```
program Ovn_2_3;  
begin  
  WriteLn('*****');  
  WriteLn('* CHICAGO *');
```

WriteLn('*****');

```
end.
```

```
program Ovn_2_4;  
const  
  konstant = 'the U.S.A.';  
begin  
  WriteLn('Born in ', konstant);  
  WriteLn('I was born in ', konstant);  
  WriteLn('I was born in ', konstant);  
  WriteLn('Born in ', konstant);
```

```
end.
```

```
program Ovn_2_5;  
var  
  tal1, tal2, summa: Integer;  
begin  
  Write('Ange första talet: '); ReadLn(tal1);  
  Write('Ange andra talet: '); ReadLn(tal2);  
  summa:= tal1 + tal2;  
  WriteLn('Summan blir ', summa);
```

```
end.
```

```
program Ovn_2_6;  
var  
  tal1, tal2, summa: Real;  
begin  
  Write('Ange första talet: '); ReadLn(tal1);  
  Write('Ange andra talet: '); ReadLn(tal2);  
  summa:= tal1 + tal2;  
  WriteLn('Summan blir ', summa:1:2);
```

```
end.
```

```

program Ovn_2_7;
const
  kurs = 4.57;
var
  dm, sek: Real;
begin
  Write('Ange antal DM: '); ReadLn(dm);
  sek:= dm * kurs;
  WriteLn(dm:1:2, ' DM = ', sek:1:2, ' SEK');
end.

```

```

program Ovn_2_8;
var
  langd, volym: Real;
begin
  Write('Ange kubens sidlängd: '); ReadLn(langd);
  volym:= langd*langd*langd;
  WriteLn('Kubens volym = ', volym:1:2);
end.

```

```

program Ovn_2_9;
var
  langd, bredd, area, omkrets: Real;
begin
  Write('Ange rektangelns längd: '); ReadLn(langd);
  Write('Ange rektangelns bredd: '); ReadLn(bredd);
  area:= langd*bredd;
  omkrets:= 2*langd + 2*bredd;
  WriteLn('Rektangelns area = ', area:1:2);
  WriteLn('Rektangelns omkrets = ', omkrets:1:2);
end.

```

```

program Ovn_2_10;
var
  pris: Real;
begin
  Write('Ange pris utan moms: '); ReadLn(pris);
  WriteLn('Pris inkl. moms: ', pris*1.25:1:2, 'kr');
end.

```

```

program Ovn_2_11;
const
  g = 9.81;
var
  m, h, W: Real;
begin
  Write('Ange massan (kg): '); ReadLn(m);
  Write('Ange höjden (m): '); ReadLn(h);
  W:= m*g*h;
  WriteLn('Potentiella energin = ', W:1:2, ' J');
end.

```

```
program Ovn_2_12;  
var  
    tal1, tal2, tal3, summa, medel: Real;  
begin  
    Write('Ange första talet: '); ReadLn(tal1);  
    Write('Ange andra talet : '); ReadLn(tal2);  
    Write('Ange tredje talet: '); ReadLn(tal3);  
    summa:= tal1 + tal2 + tal3;  
    medel:= summa/3;  
    WriteLn('Medelvärdet = ', medel:1:2);  
    WriteLn('Summan = ', summa:1:2);  
end.  
  
program Ovn_2_13;  
var  
    celsius, kelvin: Real;  
begin  
    Write('Ange temperaturen i Celsius-grader: '); ReadLn(celsius);  
    kelvin:= 273 + celsius;  
    WriteLn(celsius:1:2, ' grader Celsius = ',  
            kelvin:1:2, ' grader Kelvin');  
end.  
  
program Ovn_2_14;  
uses CRT;  
begin  
    ClrScr;  
    GotoXY(35, 12); Write('CENTRERAT');  
end.  
  
program Ovn_2_15;  
uses CRT;  
var  
    siffra: Char;  
    x, y : Integer;  
begin  
    Write('Ange en siffra: '); ReadLn(siffra);  
    WriteLn('Var på skärmen skall siffran placeras?');  
    Write('Ange x-koordinaten: '); ReadLn(x);  
    Write('Ange y-koordinaten: '); ReadLn(y);  
    ClrScr;  
    GotoXY(x, y);  
    Write(siffra);  
end.  
  
program Ovn_2_16;  
uses Printer;  
const  
    konstant = 'Tomelilla';  
begin  
    WriteLn(lst, 'Born in ', konstant);  
    WriteLn(lst, 'I was born in ', konstant);  
    WriteLn(lst, 'I was born in ', konstant);  
    WriteLn(lst, 'Born in ', konstant);  
end.
```

```
program Ovn_2_17;  
(* Denna lösning fungerar ej om det inmatade talet är >= 10 *)  
var  
    tal: Real;  
begin  
    Write('Ange ett reellt tal: '); ReadLn(tal);  
    WriteLn('Utskrift');  
    WriteLn(tal:3:1);  
    WriteLn(tal:4:1);  
    WriteLn(tal:5:1);  
    WriteLn(tal:6:1);  
    WriteLn(tal:7:1);  
end.  
  
program Ovn_2_17x;  
(* Lösning som utnyttjar möjligheten att avläsa markörpositionen *)  
uses CRT;  
  
var  
    x : Integer;  
    tal: Real;  
begin  
    Write('Ange ett reellt tal: '); ReadLn(tal);  
    WriteLn('Utskrift');  
    Write(tal:1:1);  
    x:= WhereX;  
    WriteLn;  
    WriteLn(tal:x:1);  
    WriteLn(tal:x+1:1);  
    WriteLn(tal:x+2:1);  
    WriteLn(tal:x+3:1);  
end.  
  
program Ovn_2_18;  
var  
    bruttolon, skatt, nettolon: Real;  
begin  
    Write('Ange bruttolön(kr): '); ReadLn(bruttolon);  
    Write('Ange skatt(kr): '); ReadLn(skatt);  
    nettolon:= bruttolon - skatt;  
    WriteLn('Nettolön = ', nettolon:1:2, ' kr.');
```

KAPITEL 3

```
program Ovn_3_1;
var
  taljare, namnare: Integer;
begin
  Write('Ange täljare: '); ReadLn(taljare);
  Write('Ange nämnare: '); ReadLn(namnare);
  WriteLn('Resten vid divisionen blir: ', taljare mod namnare);
  WriteLn('Heltalsdelen blir: ', taljare div namnare);
end.

program Ovn_3_2;
var
  tecken: Char;
begin
  Write('Ange ett tecken: '); ReadLn(tecken);
  WriteLn('Tecknet ', tecken, ' har ordningsnumret ', Ord(tecken));
end.

program Ovn_3_3;
var
  ordningsnr: Integer;
begin
  Write('Ange ordningsnr i ASCII-tabellen: '); ReadLn(ordningsnr);
  Write('Ordningsnr: ', ordningsnr, ' är tecknet: ', Chr(ordningsnr));
  WriteLn(' i ASCII-tabellen.');
```

```
end.

program Ovn_3_4;
var
  x, y: Char;
begin
  Write('Ange första tecknet: '); ReadLn(x);
  Write('Ange andra tecknet: '); ReadLn(y);
  Write('Att tecknet ', x, ' är större än ', y, ' är ', x>y);
end.

program Ovn_3_6;
var
  b: Boolean;
begin
  b:= true;
  WriteLn(b);
  b:= false;
  WriteLn(b);
end.

program Ovn_3_7;
var
  a, b, c: Integer;
begin
  Write('Ange första talet: '); ReadLn(a);
  Write('Ange andra talet: '); ReadLn(b);
  Write('Ange tredje talet: '); ReadLn(c);
  WriteLn('Att ', a, '<=', b, '<=', c, ' är ', (a<=b) and (b<=c));
end.
```

```

program Ovn_3_8;
var
  a, b: Boolean;
begin
  a:= false;
  b:= false;
  WriteLn('          a          b          (a and b) (a or b)  (a xor b)');
  WriteLn(a:10, b:10, a and b:10, a or b:10, a xor b:10);
  a:= false;
  b:= true;
  WriteLn(a:10, b:10, a and b:10, a or b:10, a xor b:10);
  a:= true;
  b:= false;
  WriteLn(a:10, b:10, a and b:10, a or b:10, a xor b:10);
  a:= true;
  b:= true;
  WriteLn(a:10, b:10, a and b:10, a or b:10, a xor b:10);
end.

```

```

program Ovn_3_9;
var
  x: Integer;
  b: Boolean;
begin
  Write('Ange ett heltal: '); ReadLn(x);
  b:= (2 < x) and (x <= 7);
  WriteLn('Att ', x, ' ligger i intervallet 2<', x, '<=7 är: ', b);
end.

```

```

program Ovn_3_10;
var
  x, y, z: Integer;
  b      : Boolean;
begin
  Write('Ange första heltalet: '); ReadLn(x);
  Write('Ange andra heltalet: '); ReadLn(y);
  Write('Ange tredje heltalet: '); ReadLn(z);
  b:= x mod 7 = 0;
  WriteLn('Att ', x, ' är jämnt delbart med 7 är: ', b);
  b:= z mod y <> 0;
  WriteLn('Att ', z, ' inte är delbart med ', y, ' är: ', b);
  b:= (x > y) or (x > z);
  Write('Att ', x, ' är större än något av talen ');
  WriteLn(y, ' och ', z, ' är: ', b);
  b:= (x > y) and (y > z);
  Write('Att ', x, ' är större än ', y, ', som i sin tur');
  WriteLn(' är större än ', z, ' är: ', b);
  b:= (x > y) xor (x > z);
  Write('Att ', x, ' är större än ett av talen ', y, ' och ');
  WriteLn(z, ', men inte båda samtidigt är: ', b);
end.

```

KAPITEL 4

```

program Ovn_4_1;
var tal: Real;
begin
  Write('Ange ett tal: '); ReadLn(tal);
  if tal < 0 then
    WriteLn('TALET ÄR NEGATIVT')
  else
    WriteLn('TALET ÄR POSITIVT');
end.

program Ovn_4_2;
var alder: Integer;
begin
  Write('Ange ålder: '); ReadLn(alder);
  if alder > 65 then
    WriteLn('Du är pensionerad!')
  else
    WriteLn('Du är troligen inte pensionerad!');
end.

program Ovn_4_3;
var radie, höjd: Real;
begin
  Write('Ange radien: '); ReadLn(radie);
  if radie >= 0 then begin
    Write('Ange höjden: '); ReadLn(höjd);
    WriteLn('Cylindervolymen = ', Pi*radie*radie*höjd:1:2);
  end
  else WriteLn('Kan ej beräkna cylindervolymen pga negativ radie!');
end.

program Ovn_4_4;
var alder, intrade: Integer;
begin
  WriteLn('VÄLKOMMEN TILL NÖJESPARKEN. ');
  Write('Hur gammal är du? '); ReadLn(alder);
  if alder < 3 then
    intrade:= 0
  else if alder < 11 then
    intrade:= 10
  else
    intrade:= 20;
  WriteLn('Inträdet blir ', intrade, ' kr. ');
end.

program Ovn_4_5;
var totalt, under3, under11, intrade: Integer;
begin
  WriteLn('VÄLKOMMEN TILL NÖJESPARKEN. ');
  Write('Hur många personer ingår i familjen? '); ReadLn(totalt);
  Write('Hur många är under tre år? '); ReadLn(under3);
  Write('Hur många är under 11 år? '); ReadLn(under11);
  intrade:= under11*10 + (totalt-under3-under11)*20;
  WriteLn('Inträdet blir ', intrade, ' kr. ');
end.

```

```

program Ovn_4_6;
var totalt, under3, under11: Integer;
    intrade, belopp          : Real;
begin
    WriteLn('VÄLKOMMEN TILL NÖJESPARKEN. ');
    Write('Hur många personer ingår i familjen? '); ReadLn(totalt);
    Write('Hur många är under tre år? '); ReadLn(under3);
    Write('Hur många är under 11 år? '); ReadLn(under11);
    intrade := under11*10 + (totalt-under3-under11)*20;
    WriteLn('Inträdet blir ', intrade:1:2, ' kr. ');
    Write('Hur mycket betalar du? '); ReadLn(belopp);
    if belopp < intrade then
        WriteLn('FÖR LITET BELOPP!')
    else
        WriteLn('Du får ', belopp-intrade:1:2, ' kr tillbaka. ');
end.

program Ovn_4_7;
var pengar          : Real;
    semesterdagar: Integer;
begin
    Write('Hur mycket pengar har du? '); ReadLn(pengar);
    Write('Hur många semesterdagar har du? '); ReadLn(semesterdagar);
    if (pengar >= 14000) and (semesterdagar > 14) then
        WriteLn('Varför inte åka till Australien. ')
    else
        WriteLn('Du får nog arbeta lite till. ');
end.

program Ovn_4_8;
var manad: Integer;
begin
    repeat
        Write('Ange en månad (1-12): '); ReadLn(manad);
    until (1 <= manad) and (manad <= 12);
end.

program Ovn_4_9;
var heltal: Integer;
begin
    Write('Ange ett heltal: '); ReadLn(heltal);
    if heltal mod 25 = 0 then
        WriteLn(heltal, ' är delbart med 25. ')
    else
        WriteLn(heltal, ' är inte delbart med 25. ');
end.

program Ovn_4_10;
var tal1, tal2: Integer;
begin
    Write('Ange första heltalet: '); ReadLn(tal1);
    Write('Ange andra heltalet: '); ReadLn(tal2);
    if tal1 mod tal2 = 0 then
        WriteLn(tal1, ' är jämnt delbart med ', tal2)
    else
        WriteLn(tal1, ' är inte jämnt delbart med ', tal2);
end.

```

```

program Ovn_4_11;
var heltal, nr: Integer;
begin
  Write('Ange ett heltal: '); ReadLn(heltal);
  Write('Nedräknat:');
  for nr:= heltal downto 1 do Write(' ', nr);
  WriteLn;
end.

program Ovn_4_12;
var rantesats, kapital: Real;
    ar
      : Integer;
begin
  Write('Hur stor är räntesatsen? '); ReadLn(rantesats);
  Write('Hur stort är startkapitalet? '); ReadLn(kapital);
  WriteLn('År          Kapital');
  for ar:= 1 to 5 do begin
    kapital:= kapital + rantesats*kapital/100;
    WriteLn(ar, kapital:21:2);
  end;
end.

program Ovn_4_13;
var tabell, i, maxtal: Integer;
begin
  Write('Vilken multiplikationstabell önskas? '); ReadLn(tabell);
  Write('Hur långt skall jag räkna? '); ReadLn(maxtal);
  WriteLn(tabell, ':ans multiplikationstabell. ');
  for i:= 1 to maxtal do
    WriteLn(tabell, ' * ', i, ' = ', tabell*i);
end.

program Ovn_4_14;
const stopp = 12345;
var manad: Integer;
begin
  Write('Ange en siffra: '); ReadLn(manad);
  while manad <> stopp do begin
    case manad of
      1 : WriteLn('Detta motsvaras av månaden JANUARI');
      2 : WriteLn('Detta motsvaras av månaden FEBRUARI');
      3 : WriteLn('Detta motsvaras av månaden MARS');
      4 : WriteLn('Detta motsvaras av månaden APRIL');
      5 : WriteLn('Detta motsvaras av månaden MAJ');
      6 : WriteLn('Detta motsvaras av månaden JUNI');
      7 : WriteLn('Detta motsvaras av månaden JULI');
      8 : WriteLn('Detta motsvaras av månaden AUGUSTI');
      9 : WriteLn('Detta motsvaras av månaden SEPTEMBER');
     10: WriteLn('Detta motsvaras av månaden OKTOBER');
     11: WriteLn('Detta motsvaras av månaden NOVEMBER');
     12: WriteLn('Detta motsvaras av månaden DECEMBER');
    else WriteLn('Någon sådan manad finns ej!');
    end (* case *);
  WriteLn('Jag avslutar programmet om du anger talet ', stopp, '.');
  Write('Ange en siffra: '); ReadLn(manad);
end;
end.

```

```

program Ovn_4_15;
var
  manad: Integer;
begin
  Write('Ange en månad (1 - 12): '); ReadLn(manad);
  case manad of
    1,3,5,7,
    8,10,12 : WriteLn('Månaden ', manad, ' innehåller 31 dagar. ');
    4,6,9,11: WriteLn('Månaden ', manad, ' innehåller 30 dagar. ');
    2       : WriteLn('Månaden 2 innehåller 28 eller 29 dagar. ');
  else
    WriteLn('Felaktig månad!');
  end;
end.

program Ovn_4_16;
uses CRT;
var
  alternativ: Integer;
  moms, pris: Real;
begin
  Write('Ange momsens i procent: '); ReadLn(moms);
  ClrScr;
  WriteLn('***** MENY *****');
  WriteLn('1. Inmatning av pris utan moms');
  WriteLn('2. Inmatning av pris med moms');
  Write('Ange alternativ 1 eller 2: '); ReadLn(alternativ);
  case alternativ of
    1: begin
      Write('Ange pris utan moms: '); ReadLn(pris);
      WriteLn('Pris inkl moms = ', pris+pris*moms/100:1:2, ' kr. ');
    end;
    2: begin
      Write('Ange pris med moms: '); ReadLn(pris);
      Write('Andelen moms av priset ', pris:1:2, ' kr = ');
      WriteLn(pris*moms/100/(1 + moms/100):1:2, ' kr. ');
    end;
  end;
end.

program Ovn_4_17;
var
  antalNegativa, antalPositiva, heltal: Integer;
begin
  antalNegativa:= 0; antalPositiva:= 0;
  repeat
    Write('Ange ett heltal (0 avslutar): '); ReadLn(heltal);
    if heltal < 0 then
      antalNegativa:= antalNegativa + 1
    else if heltal > 0 then
      antalPositiva:= antalPositiva + 1;
  until heltal = 0;
  WriteLn;
  WriteLn('Antalet positiva heltal: ', antalPositiva);
  WriteLn('Antalet negativa heltal: ', antalNegativa);
end.

```

```

program Ovn_4_18;
var nummer: Integer;
begin
  Write('Ange en månadsnummer (1-12): '); ReadLn(nummer);
  case nummer of
    1: WriteLn('Månaden JANUARI har 31 dagar. ');
    2: WriteLn('Månaden FEBRUARI har 28 eller 29 dagar. ');
    3: WriteLn('Månaden MARS har 31 dagar. ');
    4: WriteLn('Månaden APRIL har 30 dagar. ');
    5: WriteLn('Månaden MAJ har 31 dagar. ');
    6: WriteLn('Månaden JUNI har 30 dagar. ');
    7: WriteLn('Månaden JULI har 31 dagar. ');
    8: WriteLn('Månaden AUGUSTI har 31 dagar. ');
    9: WriteLn('Månaden SEPTEMBER har 30 dagar. ');
    10: WriteLn('Månaden OKTOBER har 31 dagar. ');
    11: WriteLn('Månaden NOVEMBER har 30 dagar. ');
    12: WriteLn('Månaden DECEMBER har 31 dagar. ');
  else
    WriteLn('Felaktigt månadsnummer!');
  end;
end.

```

```

program Ovn_4_19;
var heltal, siffra, omvant: Integer;
begin
  Write('Ange ett 3-siffrigt tal: '); ReadLn(heltal);
  omvant:= 0;
  for siffra:= 1 to 3 do begin
    omvant:= 10*omvant + heltal mod 10;
    heltal:= heltal div 10;
  end;
  WriteLn('Siffrorna i omvänd ordning: ', omvant);
end.

```

```

program Ovn_4_20;
var
  heltal, kopia, siffra: Integer;
  omvant, produkt, test: LongInt;
begin
  for heltal:= 100 to 999 do begin
    omvant:= 0; kopia:= heltal;
    for siffra:= 1 to 3 do begin
      omvant:= 10*omvant + kopia mod 10;
      kopia := kopia div 10;
    end;
    produkt:= heltal*omvant;
    test:= 10; (* Kvadrat måste vara minst 100 *)
    while test*test < produkt do
      test:= test + 1;
    if test*test = produkt then begin
      if omvant < 10 then Write('00')
      else if omvant < 100 then Write('0');
      WriteLn(omvant, '*', heltal, ' = ', produkt, ' är en kvadrat');
    end;
  end;
end.

```

```

program Ovn_4_21;
var n, summa, term: Integer;
begin
  WriteLn('Programmet beräknar seriesumman: 5+6+7+8+...+n');
  WriteLn;
  Write('Ange ett värde på n: '); ReadLn(n);
  if n >= 5 then begin
    summa:= 0;
    for term:= 5 to n do summa:= summa + term;
    WriteLn('Seriesumman = ', summa);
  end
  else
    WriteLn('Värdet på n måste vara minst 5.');
```

end.

```

program Ovn_4_22;
var k, m: Real;
begin
  WriteLn('En linje kan beskrivas med ekvationen: y=kx+m');
  Write('Ange ett värde på k: '); ReadLn(k);
  Write('Ange ett värde på m: '); ReadLn(m);
  WriteLn;
  WriteLn('Linjen: y=', k:1:1, 'x+', m:1:1, ' skär:');
  WriteLn('x-axeln i punkten: (', -m/k:1:1, ', 0)');
  WriteLn('y-axeln i punkten: (0, ', m:1:1, ')');
```

end.

```

program Ovn_4_23;
var tottid, tim, min, sek: Integer;
begin
  Write('Ange totaltid i sekunder: '); ReadLn(tottid);
  tim:= tottid div 3600;
  min:= (tottid mod 3600) div 60;
  sek:= (tottid mod 3600) mod 60;
  WriteLn;
  WriteLn(tottid, ' sek = ', tim, ' tim ', min, ' min ', sek, ' sek');
```

end.

```

program Ovn_4_24;
var antal, term, summa: LongInt;
begin
  WriteLn('Programmet summerar termerna i serien:');
  WriteLn(' 1 + 4 + 9 + 16 + 25 + 36 + 64 + ...');
  WriteLn('tills summan överstiger 1 000 000');
  antal:= 0;
  summa:= 0;
  term := 1;
  while summa <= 1000000 do begin
    summa:= summa + Sqr(term);
    antal:= antal + 1;
    term := term + 1;
  end;
  WriteLn('Summan av de ', antal, ' första termerna är ', summa);
```

end.

```
program Ovn_4_25;
var
  begonia, fuchsia: Integer;
begin
  begonia:= 8;
  repeat
    begonia:= begonia + 1;
    fuchsia:= 0;
    repeat
      fuchsia:= fuchsia + 1;
    until (fuchsia = begonia + 1) or
      ((begonia+1)*(fuchsia+1) + 2*(begonia+2)*(fuchsia+2) = 1382);
  until (begonia = 30) or
    ((begonia+1)*(fuchsia+1) + 2*(begonia+2)*(fuchsia+2) = 1382);
  WriteLn('Askungen är ', begonia-8, ' år.');
```

```
  WriteLn('Begonia är ', begonia, ' år.');
```

```
  WriteLn('Fuchsia är ', fuchsia, ' år.');
```

```
end.
```

KAPITEL 5

```
program Ovn_5_1;

  procedure Utskrift;
  begin
    WriteLn('SOLEN SKINER!');
  end;

begin
  Utskrift;
end.
```



```
program Ovn_5_2;

  procedure Utskrift;
  begin
    WriteLn('SOLEN SKINER!');
  end;

var
  klockan: Real;
begin
  Write('Ange klockan i formen tim.min: '); ReadLn(klockan);
  if (klockan >= 6.0) and (klockan <= 21.0) then
    Utskrift
  else
    WriteLn('Antingen är klockan före 6.00 eller efter 21.00');
end.
```



```
program Ovn_5_3;

var tal: Integer;

  procedure Raknare;
  begin
    tal:= 1;
    WriteLn('Uppräkning av talen ett till och med fem. ');
    while tal < 6 do begin
      WriteLn(tal);
      tal:= tal+1;
    end;
    WriteLn('PROCEDUREN Raknare AVSLUTAD. ');
  end;

begin
  Raknare;
end.
```

```
program Ovn_5_4;
```

```
  procedure Version(x: Integer);
```

```
  begin
```

```
    case x of
```

```
      3..6: WriteLn('Du har Turbo Pascal ver ', x,  
                  '.0, uppgradera snarast');
```

```
      7   : WriteLn('Du kan lugnt fortsätta och programmera');
```

```
    else
```

```
      WriteLn('Okänd version');
```

```
    end;
```

```
  end;
```

```
var
```

```
  tal: Integer;
```

```
begin
```

```
  Write('Ange Turbo-version: '); ReadLn(tal);
```

```
  Version(tal);
```

```
End.
```

```
program Ovn_5_7;
```

```
  procedure OmvSek(totaltid: Integer);
```

```
  var
```

```
    tim, min, sek: Integer;
```

```
  begin
```

```
    tim:= totaltid div 3600;
```

```
    min:= (totaltid mod 3600) div 60;
```

```
    sek:= (totaltid mod 3600) mod 60;
```

```
    WriteLn(tim, ' tim ', min, ' min ', sek, ' sek');
```

```
  end;
```

```
var
```

```
  tid: Integer;
```

```
begin
```

```
  Write('Ange totaltid i sekunder: '); ReadLn(tid);
```

```
  WriteLn;
```

```
  OmvSek(tid);
```

```
end.
```

```
program Ovn_5_8;
```

```
  procedure Omvandla(pris: Real);
```

```
  begin
```

```
    Write('Priset ', Int(pris):1:0, ':', 100*Frac(pris):1:0, ' kr');
```

```
  end;
```

```
var
```

```
  pris: Real;
```

```
begin
```

```
  Write('Ange pris i decimalform: '); ReadLn(pris);
```

```
  Omvandla(pris);
```

```
end.
```

```

program Ovn_5_9;
uses CRT;

procedure Arbete(frekv: Integer);
begin
  repeat
    GotoXY(33, 12);
    Write('ARBETE PÅGÅR!');
    Delay(frekv);
    GotoXY(33, 12);
    Write(' ');
    Delay(frekv);
  until KeyPressed;
end;

begin
  ClrScr;
  Arbete(500);
end.

program Ovn_5_10;
uses CRT;

procedure Meny(var ch: Char);
begin
  repeat
    ClrScr;
    GotoXY(34, 5); Write('MENY');
    GotoXY(34, 6); Write('****');
    GotoXY(30, 8); Write('1. Start');
    GotoXY(30, 9); Write('2. Utskrift');
    GotoXY(30,10); Write('3. Inläsning');
    GotoXY(30,11); Write('4. Ändring');
    GotoXY(30,12); Write('0. Sluta');
    GotoXY(28,14); Write('Välj alternativ: ');
    ReadLn(ch);
  until ch in ['0'..'4'];
end; (* Meny *)

var
  alt: Char;
begin
  repeat
    Meny(alt);
    GotoXY(28, 20);
    case alt of
      '1': WriteLn('Nu utförs start');
      '2': WriteLn('Nu utförs utskrift');
      '3': WriteLn('Nu utförs inläsning');
      '4': WriteLn('Nu utförs ändring');
      '0': WriteLn('Programkörningen avslutad');
    end; (* case *)
    Delay(2000);
  until alt = '0';
end.

```

```

program Ovn_5_11;

  procedure Potensform(x: Integer);
  var
    expon, heltalsdel, decimaldel, decvikt: Integer;
  begin
    expon:= 0; heltalsdel:= x; decvikt:= 1;
    while heltalsdel >= 10 do begin
      decvikt:= 10*decvikt;
      heltalsdel:= heltalsdel div 10;
      expon:= expon+1;
    end;
    decimaldel:= x mod decvikt;
    Write(heltalsdel, '.', decimaldel, '*10^', expon);
  end;

var tal: Integer;
begin
  Write('Skriv ett heltal: '); ReadLn(tal);
  Write('Talet i potensform = ');
  Potensform(tal);
end.

```

```

program Ovn_5_12;

  procedure Add(a, b: Integer; var c: Integer);
  begin c:= a + b; end;

var resultat: Integer;
begin
  Add(2, 3, resultat);
  WriteLn(resultat);
end.

```

```

program Ovn_5_13;

  procedure TakeThree(x, y, z: Integer; var ok: Boolean);
  begin ok:= (x < y) and (y < z); end;

var ok: Boolean;
begin
  TakeThree(3, 5, 4, ok);
  WriteLn('3<5<4 är ', ok);
end.

```

```

program Ovn_5_16;

  function AdderaTre(a, b, c: Integer): Integer;
  begin AdderaTre:= a + b + c; end;

var tal1, tal2, tal3: Integer;
begin
  Write('Skriv tre heltal: '); ReadLn(tal1, tal2, tal3);
  WriteLn('Summan = ', AdderaTre(tal1, tal2, tal3));
end.

```

```
program Ovn_5_17;
```

```
    function Nyvaluta(kurs, sek: Real): Real;
    begin Nyvaluta:= sek/kurs; end;
```

```
var sek, kurs: Real;
```

```
begin
```

```
    Write('Ange kursen: '); ReadLn(kurs);
    Write('Ange krontalet: '); ReadLn(sek);
    WriteLn('I nya valutan är det ', Nyvaluta(kurs, sek):1:2);
```

```
end.
```

```
program Ovn_5_18;
```

```
    function Kubik(x: Real): Real;
    begin Kubik:= x*x*x; end;
```

```
var tal: Real;
```

```
begin
```

```
    Write('Ange ett tal: '); ReadLn(tal);
    WriteLn('Talets kubik = ', Kubik(tal):1:1);
```

```
end.
```

```
program Ovn_5_19;
```

```
    function Lika(a, b: Integer): Boolean;
    begin Lika:= a = b; end;
```

```
var tall, tal2: Integer;
```

```
begin
```

```
    Write('Skriv ett heltal: '); ReadLn(tall);
    Write('Skriv ett heltal till: '); ReadLn(tal2);
    WriteLn(tall, ' = ', tal2, ' är ', Lika(tall, tal2));
```

```
end.
```

```
program Ovn_5_20;
```

```
    function Sgd(a, b: Integer): Integer;
```

```
    var r: Integer;
```

```
    begin
```

```
        r:= a mod b;
```

```
        while r <> 0 do begin
```

```
            a:= b; b:= r; r:= a mod b;
```

```
        end;
```

```
        Sgd:= b;
```

```
    end;
```

```
var t1, t2: Integer;
```

```
begin
```

```
    Write('Ange första heltalet: '); ReadLn(t1);
```

```
    Write('Ange andra heltalen : '); ReadLn(t2);
```

```
    WriteLn('Största delare till ', t1, ' och ', t2, ' = ', Sgd(t1,t2));
```

```
end.
```

```
program Ovn_5_21;
```

```

function Sign(number: Real): Integer;
begin
  if number < 0 then
    Sign:= -1
  else if number > 0 then
    Sign:= 1
  else
    Sign:= 0;
end;

```

```
var tal: Real;
```

```

begin
  Write('Skriv ett tal: '); ReadLn(tal);
  WriteLn('Sign ger ', Sign(tal));
end.

```

```
program Ovn_5_22;
```

```

function CirkelArea(radie: Real): Real;
begin CirkelArea:= Pi*radie*radie; end;

```

```
var radie: Real;
```

```

begin
  Write('Ange cirkelradien: '); ReadLn(radie);
  WriteLn('Arean = ', CirkelArea(radie):1:2);
end.

```

```
program Ovn_5_23;
```

```

function Hyp(kat1, kat2: Real): Real;
begin Hyp:= Sqrt(Sqr(kat1)+Sqr(kat2)); end;

```

```
var katet1, katet2: Real;
```

```

begin
  WriteLn('Beräkning av hypotenusan');
  Write('Ange första katetens längd: '); ReadLn(katet1);
  Write('Ange andra katetens längd: '); ReadLn(katet2);
  WriteLn('Hypotenusan = ', Hyp(katet1, katet2):1:2);
end.

```

```
program Ovn_5_24;
```

```

function Max(a, b: Real): Real;
begin if a > b then Max:= a else Max:= b; end;

```

```
var x, y: Real;
```

```

begin
  Write('Ange första talet: ');
  ReadLn(x);
  Write('Ange andra talet: ');
  ReadLn(y);
  WriteLn('Största talet är ', Max(x, y):1:2);
end.

```

```
program Ovn_5_25;
```

```
    function Linekv(a, b: Real): Real;
    begin Linekv:= -b/a; end;
```

```
var a, b: Real;
```

```
begin
```

```
    WriteLn('Ekvationen a*x + b = 0');
    Write('Ange a: '); ReadLn(a);
    Write('Ange b: '); ReadLn(b);
    WriteLn('Lösning: x = ', Linekv(a,b):1:2);
```

```
end.
```

```
program Ovn_5_26;
```

```
var
```

```
    heltal: Integer;
```

```
begin
```

```
    Write('Ange ett heltal: '); ReadLn(heltal);
    WriteLn('Kvadraten = ', Sqr(heltal));
    if heltal < 0 then
        WriteLn('Kan ej beräkna roten. Talet är negativt.')
    else
        WriteLn('Roten = ', Sqrt(heltal):1:2);
```

```
end.
```

```
program Ovn_5_27;
```

```
var p, q, d, x, x1, x2, re, im: Real;
```

```
begin
```

```
    WriteLn('Programmet beräknar rötterna till andragradsekvationen:');
    WriteLn;
```

```
    WriteLn('          x2 + px + q = 0');
    WriteLn;
```

```
    Write('Ange värde på p: '); ReadLn(p);
    Write('Ange värde på q: '); ReadLn(q);
    WriteLn;
```

```
    d:= p*p - 4*q;
```

```
    if d=0 then begin
```

```
        x:= -p/2;
```

```
        WriteLn('Ekvationen har dubbelroten x = ', x:1:2);
```

```
    end
```

```
    else if d>0 then begin
```

```
        if p<0 then
```

```
            x1:= (-p + Sqrt(d))/2
```

```
        else
```

```
            x1:= (-p - Sqrt(d))/2;
```

```
        x2:= q/x1;
```

```
        WriteLn('Ekvationen har 2 reella rötter: x1 = ', x1:1:2);
```

```
        WriteLn('          x2 = ', x2:1:2);
```

```
    end
```

```
    else begin
```

```
        WriteLn('Ekvationen saknar reella rötter!');
```

```
        re:= -p/2; im:= Sqrt(-d)/2;
```

```
        WriteLn('De komplexa rötterna är: x1 = ',re:1:2,' + ',im:1:2,'i');
```

```
        WriteLn('          x2 = ',re:1:2,' - ',im:1:2,'i');
```

```
    end;
```

```
end.
```

```

program Ovn_5_28;
var x1, x2, y1, y2, avstand: Real;
begin
  WriteLn('Ange värde på punkt 1:');
  Write('x = '); ReadLn(x1);
  Write('y = '); ReadLn(y1);
  WriteLn;
  WriteLn('Ange värde på punkt 2:');
  Write('x = '); ReadLn(x2);
  Write('y = '); ReadLn(y2);
  WriteLn;
  avstand:= Sqrt(Sqr(x1-x2) + Sqr(y1-y2));
  WriteLn('Avståndet mellan punkterna är: ', avstand:1:2);
end.

```

```

program Ovn_5_29;
var radianer: Real;
begin
  WriteLn('      x      sin(x)      cos(x)');
  WriteLn('-----');
  WriteLn;
  radianer:= 0;
  while radianer <= Pi/2 do begin
    WriteLn(radianer:6:1, Sin(radianer):11:2, Cos(radianer):11:2);
    radianer:= radianer + 0.1;
  end;
end.

```

```

program Ovn_5_30;
var x, y: Real;
begin
  WriteLn('Funktionen y = x*x*x - 6*x*x + 4'); WriteLn;
  WriteLn('      x      y'); WriteLn;
  x:= -2;
  while x <= 6 do begin
    y:= x*x*x - 6*x*x + 4;
    WriteLn(x:7:1, y:12:4);
    x:= x + 0.5;
  end;
end.

```

```

program Ovn_5_31;
var term, summa, antal: Integer;
begin
  term := 0; summa:= 0; antal:= 0;
  while summa <= 5000 do begin
    term := term + 1;
    summa:= summa + Sqr(term);
    antal:= antal + 1;
  end;
  WriteLn('Det behövs ', antal, ' termer i serien:');
  WriteLn('12 + 22 + 32 + 42 + ...');
  WriteLn('för att summan skall överstiga 5000');
end.

```

```

program Ovn_5_32;
var slumpTal: Integer;
begin
  Randomize;
  for slumpTal:= 1 to 20 do
    WriteLn(Random:20:2);
end.

```

```

program Ovn_5_33;
var slumpTal: Integer;
begin
  Randomize;
  for slumpTal:= 1 to 20 do
    WriteLn(Random(32):20);
end.

```

```

program Ovn_5_34;
var antalSexor, kast, prickar: Integer;
begin
  Randomize;
  antalSexor:= 0;
  for kast:= 1 to 100 do begin
    prickar:= Random(6) + 1;
    Write(prickar: 2); (* visa resultatet *)
    if prickar = 6 then
      antalSexor:= antalSexor + 1;
    end;
    WriteLn('Antal 6:or = ', antalSexor);
end.

```

```

program Ovn_5_35;
var
  n: Integer;
begin
  n:= 0;
  while n*n*n - 3*n <= 10000 do
    n:= n + 1;
    WriteLn('n = ', n);
end.

```

```

program Ovn_5_36;
var
  n, term, nr: Integer;
  summa      : Real;
begin
  WriteLn('Summan 1 + 1/4 + 1/9 + ... + 1/n2');
  WriteLn;
  Write('Hur många termer vill du summera? '); ReadLn(n);
  term:= 0;
  summa:= 0;
  for nr:= 1 to n do begin
    term:= term + 1;
    summa:= summa + 1/Sqr(term);
  end;
  WriteLn('Summan blir ', summa :1:2);
end.

```

```

program Ovn_5_37;
var
    n          : LongInt;
    summa, term: Real;
begin
    WriteLn('Summan 1/12 + 1/22 + 1/32 + 1/42 + ...');
    WriteLn;
    WriteLn('Programmet tar bara med termer som är > 10E-6');
    WriteLn;
    n:= 0;
    summa:= 0;
    term := 1;
    while term > 1E-6 do begin
        summa:= summa + term;
        n:= n + 1;
        term:= 1/Sqr(n+1);
    end;
    WriteLn('Summan av de ', n, ' första termerna blir ', summa :1:2);
end.

```

```

program Ovn_5_38;

    function AritmetiskSumma(a1, d: Real; n: Integer): Real;
    (* Framräknad med formel *)
    begin
        AritmetiskSumma:= n *(a1 + a1 + (n-1)*d)/2;
    end;

var
    n: Integer;
    term, differens: Real;
begin
    WriteLn('Den aritmetiska summan');
    WriteLn;
    Write('Ange första termen: '); ReadLn(term);
    Write('Ange differensen : '); ReadLn(differens);
    Write('Ange antal termer : '); ReadLn(n);
    WriteLn('Summan av de ', n, ' första termerna blir ',
        AritmetiskSumma(term, differens, n) :1:2);
end.

```

Den aritmetiska summan kan även beräknas genom "råräkning":

```

function AritmetiskSumma(a1, d: Real; n: Integer): Real;
var
    summa: Real;
    i     : Integer;
begin
    summa:= a1;
    for i:= 2 to n do begin
        a1:= a1+ d;
        summa:= summa + a1;
    end;
    AritmetiskSumma:= summa;
end;

```

```

program Ovn_5_39;

  function Avrunda(belopp: Real): Real;
  var temp: Real;
  begin
    Avrunda:= Int(2*belopp+0.5) * 0.5;
  end;

var
  belopp: Real;
begin
  Write('Ange belopp: '); ReadLn(belopp);
  WriteLn('Beloppet avrundat = ', Avrunda(belopp):1:2);
end.

program Ovn_5_40;

type
  Str5 = string[5];

  procedure InmataKod(korrektkod: Str5);
  var kod: Str5;
  begin
    repeat
      Write('Slå kodtalet: '); ReadLn(kod);
    until kod = korrektkod;
  end;

  procedure InmataOktantal(var oktan: Integer);
  begin
    repeat
      Write('Vilket oktantal 96/98: '); ReadLn(oktan);
    until oktan in [96, 98];
  end;

  procedure InmataLiter(var liter: Real);
  begin
    Write('Antal liter: '); ReadLn(liter);
  end;

  function Summa(oktan: Integer; liter: Real): Real;
  var literpris: Real;
  begin
    case oktan of
      96: literpris:= 7.53;
      98: literpris:= 7.74;
    end;
    Summa:= literpris*liter;
  end;

  procedure Kvitto(oktan: Integer; liter, summa: Real);
  begin
    WriteLn('Inköp hos Paulimacken');
    WriteLn(liter:1:1, ' liter ', oktan, '-oktanig bensin');
    WriteLn('Summa ', summa:1:2, ' kr');
  end;

```

```

var
  liter: Real;
  oktan: Integer;
begin
  WriteLn('Välkommen');
  WriteLn('Stick in kort');
  InmataKod('7676');
  InmataOktantal(oktan);
  InmataLiter(liter);
  WriteLn;
  Kvitto(oktan, liter, Summa(oktan, liter));
end.

```

```

program Ovn_5_41;

  function Reverse(num: LongInt): LongInt;
  var revnum: LongInt;
  begin
    if num >= 0 then begin
      revnum:= 0;
      repeat
        revnum:= revnum*10 + (num mod 10);
        num:= num div 10;
      until num = 0;
    end
    else revnum:= -1;
    Reverse:= revnum;
  end;

  var tal: LongInt;
  begin
    Write('Skriv ett tal: '); ReadLn(tal);
    WriteLn(Reverse(tal));
  end.

```

```

program Ovn_5_42;

  function Symmetrisk(num: LongInt): Boolean;
  var revnum: LongInt;
  begin
    if num < 0 then
      num:= -num;
      revnum:= 0;
    while revnum < num do begin
      revnum:= revnum*10 + (num mod 10);
      num:= num div 10;
    end;
    Symmetrisk:= revnum = num;
  end;

  var tal: LongInt;
  begin
    Write('Skriv ett tal: '); ReadLn(tal);
    WriteLn(Symmetrisk(tal));
  end.

```

```

program Ovn_5_43a;

  function Skottar(ar: Integer): Integer;
  begin
    if ((ar mod 4 = 0) and (ar mod 100 <> 0)) or (ar mod 400 = 0) then
      Skottar:= 1
    else
      Skottar:= 0;
    end;

  var artal: Integer;
  begin
    Write('Skriv årtal: '); ReadLn(artal);
    if Skottar(artal) = 1 then
      WriteLn('Skottår')
    else
      WriteLn('Ej skottår');
  end.

```

```

program Ovn_5_44;

  function GetInteger(min, max: Integer): Integer;
  var
    ok: Boolean;
    talet: Integer;
  begin
    repeat
      Write('Ange ett heltal i intervallet ', min, '..', max, ': ');
      ReadLn(talet);
      ok:= (talet >= min) and (talet <= max);
      if not ok then WriteLn('Felaktigt tal!');
    until ok;
    GetInteger:= talet;
  end;

  var x: Integer;
  begin
    x:= GetInteger(10,100);
    WriteLn('Talet ', x, ' är OK');
  end.

```

```

program Ovn_5_45;

  procedure LasSidor(var sid1, sid2, sid3: Real);
  begin
    repeat
      Write('Ange sida 1 (cm): '); ReadLn(sid1);
    until sid1 > 0;
    repeat
      Write('Ange sida 2 (cm): '); ReadLn(sid2);
    until sid2 > 0;
    repeat
      Write('Ange sida 3 (cm): '); ReadLn(sid3);
    until sid3 > 0;
  end;

```

```

procedure SorteraSidor(var a, b, c: Real);
var
    temp: Real;
begin
    if b > a then begin
        temp:= a;
        a:= b;
        b:= temp;    (* Nu a > b *)
    end;
    if c > a then begin    (* c är störst *)
        temp:= a;
        a:= c; c:= b;
        b:= temp;
    end
    else if c > b then begin
        temp:= b;
        b:= c;
        c:= temp;
    end
end;

function Triangel(x, y, z: Real): Boolean;
begin
    Triangel:= (x+y > z) and (x+z > y) and (y+z > x);
end;

function Heronarea(x, y, z: Real): Real;
var
    p: Real;
begin
    if Triangel(x, y, z) then begin
        p:= (x+y+z)/2;
        Heronarea:= Sqrt(p*(p-x)*(p-y)*(p-z));
    end
    else
        Heronarea:= 0;
    end;

var
    sid1, sid2, sid3, area: Real;
begin
    WriteLn('Triangelarea med Herons formel. ');
    WriteLn;
    LasSidor(sid1, sid2, sid3);
    SorteraSidor(sid1, sid2, sid3);
    WriteLn(sid1:10:1, sid2:10:1, sid3:10:1);
    area:= Heronarea(sid1, sid2, sid3);
    if area > 0 then
        WriteLn('Arean = ', area:1:2, ' cm². ')
    else
        WriteLn('Dessa sidor bildar ingen triangel!');
end.

```

KAPITEL 6

Anmärkning: Sista parametern till InitGraph-proceduren anger filvägen till den drivrutin som grafikprogrammet behöver. I vissa lösningar nedan anges 'C:\BP\BGI' och i andra enbart '' (tom sträng, varvid drivrutinen förutsätts finnas i aktuell katalog). För att programmen ska fungera kan detta behöva ändras (drivrutinerna brukar ligga i katalogen BGI som är en underkatalog till Pascal-systemet).

```

program Ovn_6_1;
uses Graph;
var graphDriver, graphMode, maxX, maxY: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then Halt(1);
  maxX:= GetMaxX; maxY:= GetMaxY;
  SetBKColor(Blue);
  SetColor(Yellow);
  Line(maxX div 3, 0, maxX div 3, maxY);
  SetColor(Red);
  Line(2*maxX div 3, 0, 2*maxX div 3, maxY);
  ReadLn;
  CloseGraph;
end.

```

```

program Ovn_6_2;
uses Graph;
var graphDriver, graphMode, maxX, maxY: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then Halt(1);
  maxX:= GetMaxX; maxY:= GetMaxY;
  SetBKColor(Blue); SetColor(Yellow);
  Line(0, maxY, maxX, maxY div 2);
  Line(maxX, maxY div 2, maxX div 2, 0);
  ReadLn;
  CloseGraph;
end.

```

```

program Ovn_6_3;
uses Graph;
var graphDriver, graphMode: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then Halt(1);
  SetBKColor(Blue); SetColor(Yellow);
  MoveTo(120,170);
  LineTo(120, 50); LineTo(140, 20);
  LineTo(160, 50); LineTo(160, 170);
  LineTo(160, 120); LineTo(180, 170);
  LineTo(100, 170); LineTo(120, 120);
  ReadLn;
  CloseGraph;
end.

```

```

program Ovn_6_4;
uses Graph;
var
  graphDriver, graphMode: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then
    Halt(1);
  SetBKColor(Blue); SetColor(Yellow);
  MoveTo(0, 100);
  LineTo(800, 100);
  LineTo(100, 0);
  LineTo(-500, 500);
  ReadLn;
  CloseGraph;
end.

```

```

program Ovn_6_5;
uses CRT, Graph;
var
  graphDriver, graphMode: Integer;
  maxX, maxY,
  centX, centY,
  sida,
  xSida, (* För de fall där aspektförh <> 1 *)
  x1, x2, y1, y2, x3, y3,
  xAsp, yAsp: Word; (* Eg onödigt på VGA *)
  hojd, aspRatio: Real;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, '');
  if GraphResult <> grOK then
    Halt(1);
  maxX:= GetMaxX; maxY:= GetMaxY;
  centX:= maxX div 2;
  centY:= maxY div 2;
  sida := centY;
  hojd := Sqrt(3*Sqr(sida)/4);
  (* Följande behövs eg inte med vanlig VGA-skärm *)
  GetAspectRatio(xAsp, yAsp);
  aspRatio:= yAsp/xAsp;
  xSida:= Round(aspRatio*sida);
  x1:= centX - xSida div 2;
  x2:= x1 + xSida;
  x3:= centX;
  y1:= Round(centY + hojd/2);
  y2:= y1;
  y3:= Round(y1 - hojd);
  MoveTo(x1, y1);
  LineTo(x2, y2);
  LineTo(x3, y3);
  LineTo(x1, y1);
  repeat until KeyPressed;
  CloseGraph;
end.

```

```

program Ovn_6_6;
uses CRT, Graph;
var
  graphDriver, graphMode      : Integer;
  maxX, maxY, centX, centY,
  sida, xSida, mxSida,
  x1, y1, x2, y2, x3, y3,
  mx1, my1, mx2, my2, mx3, my3,
  xAsp, yAsp                  : Word;
  hojd, aspRatio              : Real;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then
    Halt(1);
  maxX := GetMaxX; maxY := GetMaxY;
  centX:= maxX div 2; centY:= maxY div 2;
  sida := centY;
  hojd := Sqrt(3*Sqr(sida)/4);
  GetAspectRatio(xAsp, yAsp);
  aspRatio:= yAsp/xAsp;
  xSida:= Round(aspRatio*sida);
  x1:= centX - xSida div 2;
  x2:= x1 + xSida;
  x3:= centX;
  y1:= Round(centY + hojd/2);
  y2:= y1;
  y3:= Round(y1 - hojd);
  MoveTo(x1, y1);
  LineTo(x2, y2);
  LineTo(x3, y3);
  LineTo(x1, y1);

  mx1:= x3;
  my1:= y1;
  MoveTo(mx1, my1);
  LineTo(x3, y3);

  mx2:= x2 + Round(hojd*cos(150*Pi/180));
  my2:= y2 - Round(hojd*sin(150*Pi/180));
  MoveTo(mx2, my2);
  LineTo(x2, y2);

  mx3:= x1 + Round(hojd*cos(30*Pi/180));
  my3:= y1 - Round(hojd*sin(30*Pi/180));
  MoveTo(mx3, my3);
  LineTo(x1, y1);

  ReadLn;
  CloseGraph;
end.

```

```

program Ovn_6_7;
uses CRT, Graph;
var
    graphDriver, graphMode,
    maxX, maxY, x1, y1, x2, y2, delta, color: Integer;
    ch: Char;
begin
    graphDriver:= detect;
    InitGraph(graphDriver, graphMode, '');
    if GraphResult <> grOK then Halt(1);
    maxX:= GetMaxX; maxY:= GetMaxY;
    delta:= (maxY+1) div 50;
    x1:= delta; x2:= maxX-delta;
    y1:= delta; y2:= maxY-delta;
    color:= GetMaxColor - (1 mod GetMaxColor);
    SetColor(color);
    Rectangle(0, 0, maxX, maxY);
    Rectangle(x1, y1, x2, y2);
    ch:= ReadKey;
    CloseGraph;
end.

```

```

program Ovn_6_8;
uses CRT, Graph;
var
    graphDriver, graphMode    : Integer;
    maxX, maxY, centX, centY, xSida, mxSida,
    x1, y1, x2, y2, x3, y3, hojd,
    mx2, my2, mx3, my3, xAsp, yAsp: Word;
    sida, aspRatio            : Real;
begin
    graphDriver:= detect;
    InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
    if GraphResult <> grOK then
        Halt(1);
    maxX := GetMaxX; maxY := GetMaxY;
    centX:= maxX div 2; centY:= maxY div 2;
    hojd := maxY div 4;
    sida := Sqrt(4*Sqr(hojd)/3);
    GetAspectRatio(xAsp, yAsp);
    aspRatio:= yAsp/xAsp;
    xSida:= Round(aspRatio*sida);
    x1:= centX - xSida div 2; x2:= x1 + xSida;
    x3:= centX; y1:= centY + hojd div 2;
    y2:= y1; y3:= y1 - hojd;
    MoveTo(x1, y1);
    LineTo(x3, y3);
    LineTo(x2, y2);
    mx2:= x2 + Round(hojd*cos(150*Pi/180));
    my2:= y2 - Round(hojd*sin(150*Pi/180));
    mx3:= x1 + Round(hojd*cos(30*Pi/180));
    my3:= y1 - Round(hojd*sin(30*Pi/180)) + 1;
    MoveTo(mx2, my2);
    LineTo(mx3, my3);
    ReadLn;
    CloseGraph;
end.

```

```

program Ovn_6_10;
uses CRT, Graph;
var
  graphDriver, graphMode,
  maxX, maxY, x1, y1, x2, y2, color: Integer;
  ch: Char;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, '');
  if GraphResult <> grOK then Halt(1);
  maxX:= GetMaxX; maxY:= GetMaxY;
  x1:= maxX div 4; x2:= 3*maxX div 4;
  y1:= maxY div 3; y2:= 2*maxY div 3;
  color:= GetMaxColor - (1 mod GetMaxColor);
  SetColor(color);
  Rectangle(x1, y1, x2, y2);
  repeat
    Delay(100);
    SetColor(black);
    Rectangle(x1, y1, x2, y2);
    Dec(x1); Dec(y1); Inc(x2); Inc(y2);
    SetColor(color);
    Rectangle(x1, y1, x2, y2);
  until (y1 = 4) or (x1 = 4);
  ch:= ReadKey;
  CloseGraph;
end.

```

```

program Ovn_6_11;
uses CRT, Graph;
var
  graphDriver, graphMode,
  x1, y1, x2, y2, centreX, centreY, mx, my, maxX, maxY, hojd: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then Halt(1);
  maxX:= GetMaxX; maxY:= GetMaxY;
  centreX:= maxX div 2; centreY:= maxY div 2;
  SetBKColor(black);
  hojd:= centreX;
  repeat
    x1:= centreX - hojd div 2; x2:= centreX + hojd div 2;
    y1:= centreY + hojd div 2; y2:= centreY - hojd div 2;
    Rectangle(x1, y1, x2, y2);
    mx:= x1 + (x2 - x1) div 2;
    my:= y2 + (y1 - y2) div 2;
    MoveTo(x1, my);
    LineTo(mx, y2);
    LineTo(x2, my);
    LineTo(mx, y1);
    LineTo(x1, my);
    hojd:= hojd div 2;
  until x1 = mx;
  ReadLn;
  CloseGraph;
end.

```

```

program Ovn_6_12;
uses Graph;
var
  graphDriver, graphMode, maxX, maxY, hojd: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then Halt(1);
  maxX:= GetMaxX; maxY:= GetMaxY;
  SetBKColor(Black);
  SetColor(Yellow);
  hojd:= maxY div 15;
  Rectangle(0, 0, maxX, maxY);
  Rectangle(hojd, hojd, maxX-hojd, maxY-hojd);
  SetFillStyle(solidFill, Blue);
  FloodFill(hojd div 2, hojd div 2, Yellow);
  ReadLn;
  CloseGraph;
end.

```

```

program Ovn_6_13;
uses Graph;
var
  graphDriver, graphMode: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then
    Halt(1);
  SetBKColor(Black);
  SetColor(Blue);
  Rectangle(170, 140, 470, 340);
  SetFillStyle(solidFill, Blue);
  FloodFill(180, 160, Blue);
  SetColor(Yellow);
  MoveTo(170, 220);
  LineTo(250, 220);
  LineTo(250, 140);
  LineTo(290, 140);
  LineTo(290, 220);
  LineTo(470, 220);
  LineTo(470, 260);
  LineTo(290, 260);
  LineTo(290, 340);
  LineTo(250, 340);
  LineTo(250, 260);
  LineTo(170, 260);
  LineTo(170, 220);
  SetFillStyle(solidFill, Yellow);
  FloodFill(320, 240, Yellow);
  ReadLn;
  CloseGraph;
end.

```

```
program Ovn_6_14;  
uses Graph;  
var  
  graphDriver, graphMode: Integer;  
begin  
  graphDriver:= detect;  
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');  
  if GraphResult <> grOK then  
    Halt(1);  
    SetBKColor(Green);  
    SetColor(LightBlue);  
    Rectangle(0, 0, 639, 240);  
    SetFillStyle(solidFill, LightBlue);  
    FloodFill(10, 10, LightBlue);  
  
    SetColor(Red);  
    Rectangle(220, 140, 420, 340);  
    SetFillStyle(solidFill, Red);  
    FloodFill(320, 240, Red);  
    MoveTo(220, 140);  
    LineTo(320, 40);  
    LineTo(420, 140);  
    FloodFill(320, 100, Red);  
  
    SetColor(Blue);  
    Rectangle(240, 260, 300, 320);  
    SetFillStyle(solidFill, Blue);  
    FloodFill(260, 300, Blue);  
  
    Rectangle(340, 260, 400, 320);  
    SetFillStyle(solidFill, Blue);  
    FloodFill(380, 300, Blue);  
  
    Rectangle(280, 160, 360, 220);  
    SetFillStyle(solidFill, Blue);  
    FloodFill(300, 180, Blue);  
  
    ReadLn;  
    CloseGraph;  
end.
```

```

program Ovn_6_15;
uses Graph;
var
  graphDriver, graphMode: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then
    Halt(1);
  SetBKColor(LightBlue);
  SetColor(White);

  Bar3D(140, 200, 240, 300, 25, true);
  SetLineStyle(solidLn, solidFill, normWidth);
  Ellipse(200, 191, 0, 360, 10, 3);
  FloodFill(200, 191, White);
  Ellipse(250, 222, 0, 360, 3, 10);
  FloodFill(250, 222, White);
  Ellipse(256, 260, 0, 360, 3, 10);
  FloodFill(256, 260, White);
  SetColor(LightBlue);
  SetFillStyle(solidFill, LightBlue);
  Circle(165, 225, 10);
  FloodFill(165, 225, LightBlue);
  Circle(190, 250, 10);
  FloodFill(190, 250, LightBlue);
  Circle(215, 275, 10);
  FloodFill(215, 275, LightBlue);

  SetBKColor(LightBlue);
  SetColor(White);
  SetFillStyle(solidFill, White);
  Bar3D(340, 200, 440, 300, 25, true);
  SetLineStyle(solidLn, solidFill, normWidth);
  Ellipse(400, 191, 0, 360, 10, 3);
  FloodFill(400, 191, White);
  Ellipse(450, 222, 0, 360, 3, 10);
  FloodFill(450, 222, White);
  Ellipse(456, 260, 0, 360, 3, 10);
  FloodFill(456, 260, White);
  SetColor(LightBlue);
  SetFillStyle(solidFill, LightBlue);
  Circle(365, 225, 10);
  FloodFill(365, 225, LightBlue);
  Circle(390, 250, 10);
  FloodFill(390, 250, LightBlue);
  Circle(415, 275, 10);
  FloodFill(415, 275, LightBlue);

  ReadLn;
  CloseGraph;
end.

```

```

program Ovn_6_16;
uses Graph;
var
  graphDriver, graphMode: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then
    Halt(1);
    SetBKColor(LightBlue);
    SetColor(White);
    Circle(300, 200, 100);
    Circle(300, 300, 100);
    ReadLn;
    CloseGraph;
end.

program Ovn_6_17;
uses Graph;
var
  graphDriver, graphMode: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then
    Halt(1);
    SetBKColor(LightBlue);
    SetColor(White);
    Circle(200, 200, 100);
    Circle(350, 200, 50);
    ReadLn;
    CloseGraph;
end.

program Ovn_6_18;
uses Graph;
var
  graphDriver, graphMode, centreX, radie: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then
    Halt(1);
    SetBKColor(LightBlue);
    SetColor(White);
    centreX:= 100;
    radie:= 100;
    while centreX + radie < GetMaxX do begin
      Circle(centreX, 200, radie);
      centreX:= centreX + radie + Round(0.69*radie);
      radie:= Round(0.69*radie);
    end;
    ReadLn;
    CloseGraph;
end.

```

```

program Ovn_6_19;
uses Graph;
var
    graphDriver, graphMode,
    centre1X, centre1Y, centre2X, centre2Y, centre3X, centre3Y: Integer;
begin
    graphDriver:= detect;
    InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
    if GraphResult <> grOK then
        Halt(1);
    SetBKColor(LightBlue);
    SetColor(White);
    centre1X:= 200;
    centre1Y:= 250;
    centre2X:= 350;
    centre2Y:= 250;
    centre3X:= 325;
    centre3Y:= 128;
    Circle(centre1X, centre1Y, 100);
    Circle(centre2X, centre2Y, 50);
    Circle(centre3X, centre3Y, 75);
    ReadLn;
    CloseGraph;
end.

```

```

program Ovn_6_20;
uses Graph;

procedure OlympiaRing(centreX, centreY, colour: Integer);
begin
    SetColor(colour);
    SetFillStyle(solidFill, colour);
    Circle(centreX, centreY, 90);
    Circle(centreX, centreY, 100);
    FloodFill(centreX+95, centreY, colour);
end;

var
    graphDriver, graphMode,
    centreX, centreY      : Integer;
begin
    graphDriver:= detect;
    InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
    if GraphResult <> grOK then
        Halt(1);
    SetBKColor(LightBlue);
    OlympiaRing(150, 200, White);
    OlympiaRing(300, 200, Red);
    OlympiaRing(450, 200, Yellow);
    OlympiaRing(225, 300, Green);
    Olympiarings(375, 300, Brown);
    ReadLn;
    CloseGraph;
end.

```

```

program Ovn_6_21;
uses Graph;
var
  graphDriver, graphMode,
  centreX, centreY, xradie: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then
    Halt(1);
  SetBKColor(LightBlue);
  centreX:= 320;
  centreY:= 200;
  xradie := 100;
  while centreX + xradie < GetMaxX do begin
    Ellipse(centreX, centreY, 0, 360, xradie, 50);
    xradie:= xradie + 5;
  end;
  ReadLn;
  CloseGraph;
end.

```

```

program Ovn_6_22;
uses Graph;
var
  graphDriver, graphMode,
  centreX, centreY, xradie: Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then
    Halt(1);
  SetBKColor(LightBlue);
  centreX:= 320;
  centreY:= 400;
  xradie := 200;
  while xradie > 0 do begin
    Ellipse(centreX, centreY, 0, 360, xradie, 50);
    xradie := xradie - 1;
    centreY:= centreY - 1;
  end;
  ReadLn;
  CloseGraph;
end.

```

```

program Ovn_6_24; (* Brownsk rörelse *)
uses CRT, Graph;
var
  graphDriver, graphMode,
  maxX, maxY,
  x1, x2, y1, y2,
  dx, dy, x, y,
  maxColor : Integer;

```

```

begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, '');
  if GraphResult <> grOK then
    Halt(1);
  maxX:= GetMaxX;
  maxY:= GetMaxY;
  maxColor:= GetMaxColor;

  x1:= maxX div 5; x2:= 4*x1;    (* Ger 3/5 av skärmen *)
  y1:= maxY div 5; y2:= 4*y1;
  SetColor(maxColor - (1 mod maxColor));
  Rectangle(x1-1, y1-1, x2+1, y2+1);
  SetViewPort(x1, y1, x2, y2, clipOn);
  SetColor(maxColor);
  MoveTo((x2-x1) div 2, (y2 - y1) div 2);
  Randomize;
  (* Nedan upprepas tills förflyttning utanför rutan.
     För att i stället upprepa 100 ggr används lämpligen for-sats *)
  repeat
    dx:= (-2 + Random(5));
    dy:= (-2 + Random(5));
    (* Pröva gärna med -15 + Random(31) i stället *)
    LineRel(dx, dy);
    x:= GetX; y:= GetY;
    Delay(1);
  until (x < 0) or (x > x2-x1) or (y < 0) or (y > y2-y1);
  (* tills utanför rutan *)

  repeat until KeyPressed;
  CloseGraph;
end.

```

```

program Ovn_6_25;
uses Graph;
var
  graphDriver, graphMode: Integer;
  antal                : Integer;
begin
  graphDriver:= detect;
  InitGraph(graphDriver, graphMode, 'C:\BP\BGI');
  if GraphResult <> grOK then
    Halt(1);
  SetBKColor(LightBlue);
  SetColor(White);
  for antal:= 1 to 10 do begin
{   SetTextJustify(centerText + 5*antal, centerText); }
  SetTextStyle(gothicFont, horizDir, antal + 3);
  OutText('Namn');
  end;
  ReadLn;
  CloseGraph;
end.

```

KAPITEL 7

```

program Ovn_7_1;
var
  svar: Char;
begin
  repeat
    WriteLn('Test på operatorn in');
    Write('Fler utskrifter (j/n): '); ReadLn(svar);
  until not (svar in ['J', 'j']);
end.

```

```

program Ovn_7_4;

type
  Ttotal = set of 0..99;

procedure SkrivMangd(m: Ttotal; max: Integer);
var
  i: Integer;
begin
  for i:= 1 to max do
    if i in m then Write(i:4);
  WriteLn;
end;

var
  m1, m2, m3: Ttotal;
  tal: Integer;
begin
  m1:= [];
  m2:= [];
  Randomize;
  for tal:= 1 to 10 do begin
    m1:= m1 + [Random(14)];
    m2:= m2 + [Random(14)];
  end;
  WriteLn('Mängd m1:');
  SkrivMangd(m1, 13);
  WriteLn('Mängd m2:');
  SkrivMangd(m2, 13);
  if m1 <= m2 then
    WriteLn('m1 är en delmängd av m2')
  else
    WriteLn('m1 är ingen delmängd av m2');
  m3:= m1-m2;
  WriteLn('Mängd m3 (=m1-m2):');
  SkrivMangd(m3, 13);
  m3:= m1+m2;
  WriteLn('Mängd m3 (=m1+m2):');
  SkrivMangd(m3, 13);
  m3:= m1*m2;
  WriteLn('Mängd m3 (=m1*m2):');
  SkrivMangd(m3, 13);
end.

```

KAPITEL 8

```

program Ovn_8_1;
var
  string1, string2: string[40];
begin
  Write('Mata in sträng 1 (max 40 tecken): '); ReadLn(string1);
  Write('Mata in sträng 2 (max 40 tecken): '); ReadLn(string2);
  WriteLn;
  if string1 = string2 then
    WriteLn('LIKA')
  else
    WriteLn('OLIKA');
end.

program Ovn_8_2;
var
  fornamn: string[30];
begin
  Write('Hej! Vad heter du? '); ReadLn(fornamn);
  WriteLn('Ditt namn börjar på bokstaven ', fornamn[1]);
end.

program Ovn_8_3;
var
  ordnum: Integer;
begin
  Write('Ange ett ordningsnummer i ASCII-tabellen: '); ReadLn(ordnum);
  Write('Ordningsnummer ', ordnum, ' motsvaras av tecknet ');
  WriteLn(Chr(ordnum), ' i ASCII-tabellen.');
```

end.

```

program Ovn_8_4;
var
  versal, gemen: Boolean;
  ordnum      : Integer;
begin
  versal:= false;
  gemen := false;
  repeat
    Write('Ange ett ordningsnummer i ASCII-tabellen: ');
    ReadLn(ordnum);
    case Chr(ordnum) of
      'A'..'Z',
      'Å', 'Ä', 'Ö': begin
        WriteLn('Motsvaras av en stor bokstav');
        versal:= true;
      end;
      'a'..'z',
      'å', 'ä', 'ö': begin
        WriteLn('Motsvaras av en liten bokstav');
        gemen:= true;
      end
    else
      WriteLn('Motsvaras inte av någon bokstav. Försök Igen!');
    end;
  until versal or gemen;
end.

```

```

program Ovn_8_5;
var ord: string[20];
begin
  Write('Skriv ett ord: '); ReadLn(ord);
  WriteLn('Ordet innehåller ', Length(ord), ' bokstäver. ');
end.

program Ovn_8_6;
var
  ord : string[20];
  antalVersaler, antalGemener, tecken: Integer;
begin
  antalVersaler:= 0;
  antalGemener := 0;
  Write('Skriv ett ord: '); ReadLn(ord);
  for tecken:= 1 to Length(ord) do
    case ord[tecken] of
      'A'..'Z', 'Å', 'Ä', 'Ö': antalVersaler:= antalVersaler + 1;
      'a'..'z', 'å', 'ä', 'ö': antalGemener := antalGemener + 1;
    end;
  Write('Ordet innehåller ', antalVersaler, ' stora bokstäver');
  WriteLn(' och ', antalGemener, ' små bokstäver. ');
end.

program Ovn_8_7;
var
  ordet : string[20];
  tecken: Integer;
begin
  Write('Skriv ett ord: '); ReadLn(ordet);
  for tecken:= 1 to Length(ordet) do
    Write(ordet[tecken], ' ');
  WriteLn;
end.

program Ovn_8_8;
var strang: string[20];
begin
  Write('Skriv en sträng: '); ReadLn(strang);
  WriteLn;
  Write('De tre första bokstäverna är ', strang[1], ', ');
  WriteLn(strang[2], ' och ', strang[3]);
  WriteLn('Strängens längd är ', Length(strang), ' tecken');
  WriteLn('Strängens sista bokstav är ', strang[Length(strang)]);
end.

program Ovn_8_9;
var
  ordet : string[20];
  tecken: Integer;
begin
  Write('Skriv ett ord: '); ReadLn(ordet);
  WriteLn;
  for tecken:= 1 to Length(ordet) do
    ordet[tecken]:= Chr(Ord(ordet[tecken])-2);
  WriteLn('Krypterat ord: ', ordet);
end.

```

```

program Ovn_8_10;
var
  ordet : string[20];
  tecken: Integer;
begin
  Write('Skriv in det krypterade ordet: '); ReadLn(ordet);
  WriteLn;
  for tecken:= 1 to Length(ordet) do
    ordet[tecken]:= Chr(Ord(ordet[tecken])+2);
  WriteLn('Ordet är: ', ordet);
end.

```

```

program Ovn_8_11;
var
  streng: string[20];
  tecken: Integer;
begin
  Write('Skriv en sträng: '); ReadLn(streng);
  WriteLn;
  Write('Strängen i omvänd ordning: ');
  for tecken:= Length(streng) downto 1 do
    Write(streng[tecken]);
  WriteLn;
end.

```

```

program Ovn_8_12;
var
  strang          : string;
  ordnum, platsnummer: Integer;
begin
  Write('Ange en godtycklig sträng: '); ReadLn(strang);
  Write('Ordningsnummer för tecken: '); ReadLn(ordnum);
  Write('Platsnummer i strängen   : '); ReadLn(platsnummer);
  strang[platsnummer]:= Chr(ordnum);
  WriteLn('Den nya strängen är      : ', strang);
end.

```

```

program Ovn_8_13;
var
  strang: string;
  tecken: Integer;
begin
  Write('Ange en sträng: '); ReadLn(strang);
  for tecken:= 1 to Length(strang) do
    case strang[tecken] of
      'a'..'z': strang[tecken]:= Chr(Ord(strang[tecken]) - 32);
      'å'      : strang[tecken]:= 'Å';
      'ä'      : strang[tecken]:= 'Ä';
      'ö'      : strang[tecken]:= 'Ö';
    end;
  WriteLn('Strängen med enbart versaler är: ', strang);
end.

```

```

program Ovn_8_14;
var
  tall, tal2           : string[3];
  heltall1, heltal2, felpos: Integer;
begin
  repeat
    Write('Ange heltal 1: '); ReadLn(tall);
    Val(tall, heltall1, felpos);
    if felpos <> 0 then WriteLn('Felaktigt heltal!');
  until felpos = 0;
  repeat
    Write('Ange heltal 2: '); ReadLn(tal2);
    Val(tal2, heltal2, felpos);
    if felpos <> 0 then WriteLn('Felaktigt heltal!');
  until felpos = 0;
  WriteLn;
  WriteLn(heltall1, '+', heltal2, '=', heltall1+heltal2);
end.

```

```

program Ovn_8_15;
var
  s, delstr: string[80];
  p: Integer;
begin
  Write('Ange en sträng: '); ReadLn(s);
  Write('Ange en delsträng: '); ReadLn(delstr);
  p:= Pos(delstr, s);
  if p <> 0 then
    WriteLn(delstr, ' finns i strängen ', s,
            ' och börjar i position ', p)
  else
    WriteLn(delstr, ' finns inte strängen ', s);
end.

```

```

program Ovn_8_16;
var
  personnr: string[11];
  pos      : Integer;
  ok       : Boolean;
begin
  repeat
    Write('Ange personnr: '); ReadLn(personnr);
    ok := personnr[7] = '-';
    pos:= 1;
    while (pos <= 11) and ok do
      if (personnr[pos] in ['0'..'9']) or (pos = 7) then
        pos:= pos + 1
      else
        ok:= false;
    until ok;
end.

```

```

program Ovn_8_17;
uses CRT;
var
  ch: Char;
  rad, kol: Integer;
begin
  ClrScr;
  rad:= 13; kol:= 40;
  WriteLn('Flytta markören med någon av tangenterna 1..9. ');
  WriteLn('Avsluta programmet med riktningen "0" ');
  GoToXY(kol, rad);
  repeat
    ch := ReadKey;
    case ch of
      '1': begin rad:= rad + 1; kol:= kol - 1; end;
      '2': rad:= rad + 1;
      '3': begin rad:= rad + 1; kol:= kol + 1; end;
      '4': kol:= kol - 1;
      '6': kol:= kol + 1;
      '7': begin rad:= rad - 1; kol:= kol - 1; end;
      '8': rad:= rad - 1;
      '9': begin rad:= rad - 1; kol:= kol + 1; end;
    end;
    case rad of
      0: rad:= 25;
      26: rad:= 1;
    end;
    case kol of
      0: kol:= 80;
      81: kol:= 1;
    end;
    GoToXY(1, 25);
    Write('Markören befinner sig nu i position ', rad:2, ', ', kol:2);
    GotoXY(kol, rad);
  until ch = '0'
end.

```

Övning 8.19

```

procedure LasKonto(var s: string);
(* LasKonto gör de kontroller som krävdes i uppgiften. För praktisk
användning är det dock inte tillräckligt, t ex släpper proceduren
igenom ett konto som består av 13 minustecken. Proceduren borde
alltså även se till att övriga ingående tecken är OK. *)
var
  ok: Boolean;
begin
  repeat
    Write('Ange konto: ');
    ReadLn(s);
    ok:= (Length(s) = 13) and (s[5] = '-') and (s[12] = '-');
    if not ok then
      WriteLn('Felaktigt inmatning!');
  until ok;
end;

```

```
program Ovn_8_20;
```

```
function Match: Char;
begin
  case Random(3) of
    0: Match:= '1';
    1: Match:= 'X';
    2: Match:= '2';
  end;
end;
```

```
var
  i: Integer;
begin
  WriteLn('TIPSRAD'); WriteLn; WriteLn;
  Randomize;
  for i:= 1 to 13 do WriteLn(Match);
end.
```

```
program Ovn_8_21;
```

```
type
  Str11 = string[11];

function PersOK(pnr: Str11): Boolean;
var
  i : Integer;
  ok: Boolean;
begin
  if Length(pnr) <> 11 then (* Rätt längd ? *)
    PersOK:= false
  else if Pos('-', pnr) <> 7 then (* "-" på rätt ställe ? *)
    PersOK:= false
  else begin
    Delete(pnr, 7, 1); (* Eliminera - *)
    (* Kontrollera att nu enbart siffror återstår *)
    i:= 1; ok:= true;
    while ok and (i <= 10) do
      if not (pnr[i] in ['0'..'9']) then
        ok:= false
      else
        i:= i+1;
        PersOK:= ok;
    end;
  end;
end;
```

```
var
  pnr: Str11;
begin (* Test *)
  Write('Skriv ett personnummer: '); ReadLn(pnr);
  WriteLn(PersOK(pnr));
end.
```

```

program Ovn_8_21x;
(* Denna lösning är fullständigare än den som krävs i uppgiften.
   Kontrollerar även datum (dock ej skottår) och kontrollsiffra *)

type
  Str11 = string[11];

function PersOK(pnr: Str11): Boolean;
var
  i, month, day, sum, siffra, kontrollsiffra: Integer;
  ok: Boolean;
begin
  if Length(pnr) <> 11 then (* Rätt längd ? *)
    ok:= false
  else if Pos('-', pnr) <> 7 then (* "-" på rätt ställe ? *)
    ok:= false
  else begin
    Delete(pnr, 7, 1); (* Eliminera - *)
    pnr:= pnr+' '; (* "vaktpost" *)
    i:= 1; (* Kontrollera att nu enbart siffror återstår *)
    while (i <= 10) and (pnr[i] in ['0'..'9']) do
      i:= i+1;
    ok:= i = 11;
    if ok then begin (* månad ok? *)
      month:= 10*(Ord(pnr[3])-Ord('0'))+Ord(pnr[4])-Ord('0');
      ok:= month <= 12;
    end; (* if *)
    if ok then begin (* dagnr ok? - struntar i skottårskoll *)
      day:= 10*(Ord(pnr[5])-Ord('0'))+Ord(pnr[6])-Ord('0');
      case month of
        4, 6, 9, 11: ok:= day <= 30;
        2: ok:= day <= 29
      else
        ok:= day <= 31;
      end; (* case *)
    end; (* if *)
    if ok then begin (* Kontrollsiffra ok? *)
      sum:= 0;
      for i:= 1 to 9 do begin
        siffra:= Ord(pnr[i])-Ord('0');
        if i mod 2 <> 0 then siffra:= 2*siffra;
        sum:= sum+siffra div 10+siffra mod 10;
      end; (* for *)
      kontrollsiffra:= (10-sum mod 10) mod 10;
      ok:= kontrollsiffra = Ord(pnr[10])-Ord('0');
    end; (* if *)
    end; (* if *)
    PersOK:= ok;
  end;

var
  pnr: Str11;
begin (* Test *)
  Write('Skriv ett personnummer: '); ReadLn(pnr);
  WriteLn(PersOK(pnr))
end.

```

```

program Ovn_8_22;

  function Yes(question: string): Boolean;
  var
    ch: Char;
  begin
    repeat
      Write(question);
      ReadLn(ch);
      ch:= UpCase(ch);
    until ch in ['N', 'J'];
    Yes:= ch = 'J';
  end;

begin (* Test *)
  if Yes('Vill du sluta? ') then
    WriteLn('Tack för idag.')
  else
    WriteLn('Då fortsätter vi.');
```

```

end.

program Ovn_8_23;

  procedure Kryptera(var s: string);
  var
    i: Integer;
  begin
    for i:= 1 to Length(s) do
      if s[i] in ['A'..'Z', 'Å', 'Ä', 'Ö',
                  'a'..'z', 'å', 'ä', 'ö', '0'..'9'] then
        s[i]:= Chr(Ord(s[i]) + 2);
    end;

  var
    s: string;
  begin
    WriteLn('KRYPTERAD TEXT');
    WriteLn;
    Write('Skriv en text: ');
    ReadLn(s);
    Kryptera(s);
    WriteLn(s);
  end.

```

```

program Ovn_8_24;

procedure Konvertera(var s: string);
var
  i : Integer;
  ch: Char;
  nystr: string;
begin
  nystr:= '';
  for i:= 1 to Length(s) do begin
    ch:= s[i];
    case ch of
      'å': nystr:= nystr + 'aa';
      'Å': nystr:= nystr + 'AA';
      'ä': nystr:= nystr + 'ae';
      'Ä': nystr:= nystr + 'AE';
      'ö': nystr:= nystr + 'oe';
      'Ö': nystr:= nystr + 'OE';
    else
      nystr:= nystr + ch;
    end;
  end;
  s:= nystr;
end;

var s:string;
begin
  Write('Ange en sträng: ');
  ReadLn(s);
  Konvertera(s);
  WriteLn('Med "nya" åäö: ', s);
end.

```

```

program Ovn_8_25;

procedure Bakvand(var s: string);
var
  i, len: Integer;
  ch: Char;
begin
  len:= Length(s);
  for i:= 1 to len div 2 do begin
    ch:= s[i];
    s[i]:= s[len-i+1];
    s[len-i+1]:= ch;
  end;
end;

var s: string;
begin
  Write('Skriv en sträng (max 80 tkn): '); ReadLn(s);
  Bakvand(s);
  WriteLn(s);
end.

```

```
program Ovn_8_26;
```

```

function Palin(s: string): Boolean;
var pal: Boolean;
    i, len, mitt: Integer;
begin
    len := Length(s); mitt:= len div 2; i:= 1;
    repeat
        pal:= s[i] = s[len+1-i];
        i:= i+1;
    until (i > mitt) or not pal;
    Palin:= pal;
end; (* Palin *)

```

```

var s: string;
begin
    Write('Skriv en sträng: '); ReadLn(s);
    if not Palin(s) then Write('ICKE ');
    WriteLn('palindrom');
end.

```

```
program Ovn_8_27;
```

```

procedure Insert(delstr: string; var strang: string; pos: Integer);
var i: Integer;
    s: string;
begin
    s:= strang; strang:= '';
    for i:= 1 to pos-1 do
        strang:= strang + s[i];
    strang:= strang + delstr;
    for i:= pos to Length(s) do
        strang:= strang + s[i];
end;

```

```

var s: string;
begin
    s:= 'Lina';
    Insert('ivl', s, 2);
    WriteLn(s);
end.

```

```
program Ovn_8_28;
```

```

procedure Delete(var strang: string; pos, antal: Integer);
var i: Integer;
    s: string;
begin
    s:= strang; strang:= '';
    for i:= 1 to pos-1 do
        strang:= strang + s[i];
    for i:= pos+antal to Length(s) do
        strang:= strang + s[i];
end;

```

```

var s: string;
begin
  s:= 'harpalt';
  Delete(s, 2, 3);
  WriteLn(s);
end.

program Ovn_8_29;

type Str80 = string[80];

procedure Tecken(s1, s2: Str80);
var
  engChars: set of 'a'..'z';
  i : Integer;
  tkn: Char;
begin
  engChars:= [];
  for i:= 1 to Length(s1) do
    if s1[i] in ['a'..'z'] then engChars:= engChars + [s1[i]];
  for i:= 1 to Length(s2) do
    if s2[i] in ['a'..'z'] then engChars:= engChars + [s2[i]];
  for tkn:= 'a' to 'z' do
    if tkn in engChars then Write(tkn);
  WriteLn;
end; (* Tecken *)

var s1, s2: Str80;
begin
  Write('Skriv en sträng: '); ReadLn(s1);
  Write('Skriv en sträng till: '); ReadLn(s2);
  Tecken(s1, s2);
end.

program Ovn_8_30;

type Str80 = string[80];

procedure GetIntegerString(var s: Str80);
var stringOK: Boolean;
  i, strlen: Integer;
  tkn: Char;
begin
  repeat
    Write('Ange ett heltal (max 8 siffror): '); ReadLn(s);
    strlen:= Length(s);
    stringOK:= (strlen > 0) and (strlen < 9);
    i:= 1;
    while stringOK and (i <= strlen) do begin
      stringOK:= s[i] in ['0'..'9'];
      i:= i + 1;
    end;
    if not stringOK then WriteLn('Felaktigt tal!');
  until stringOK;
end; (* GetIntegerString *)

```

```

var
  talstr: Str80;
begin
  GetIntegerString(talstr);
  WriteLn('Den inlästa strängen är ', talstr);
end.

program Ovn_8_31; (* Enklare variant. Ingen felhantering utförs *)
type
  Str80 = string[80];

  function StrToInt(s: Str80): LongInt;
  (* Förutsätter att s innehåller max 8 siffertecken *)
  var
    heltal: LongInt;
    i: Integer;
  begin
    heltal:= 0;
    for i:= 1 to Length(s) do
      heltal:= heltal*10 + Ord(s[i]) - Ord('0');
      StrToInt:= heltal;
    end;

var
  talstr: Str80;
begin
  WriteLn(StrToInt('53'));
  WriteLn(5*StrToInt('101'));
end.

program Ovn_8_32;

  procedure AreYouQuitting(var yes: Boolean);
  var
    svar: Char;
  begin
    repeat
      WriteLn;
      Write('Vill du sluta? '); ReadLn(svar);
      case svar of
        'j', 'J': yes:= true;
        'n', 'N': yes:= false;
      else
        WriteLn('Felaktigt svar!');
      end;
    until svar in ['j', 'J', 'n', 'N'];
  end;

var quit: Boolean;
begin
  repeat
    WriteLn;
    WriteLn('Testutskrift');
    AreYouQuitting(quit);
  until quit;
end.

```

KAPITEL 9

```

program Ovn_9_1;
var
  filnamn, textrad: string;
  textfil: Text;
begin
  Write('Ange textfilens namn: '); ReadLn(filnamn);
  Assign(textfil, filnamn); Reset(textfil);
  while not EOF(textfil) do begin
    ReadLn(textfil, textrad);
    WriteLn(textrad);
  end;
  Close(textfil);
end.

program Ovn_9_2;
type
  Frekvenstabell = array[1..5] of Integer;
var
  filnamn: string;
  fil : Text;
  summa: Real;
  frekv: Frekvenstabell;
  antal, betyg: Integer;
begin
  Write('Ange textfilens namn (med filväg): ');
  ReadLn(filnamn);
  Assign(fil, filnamn);
  Reset(fil);
  for betyg:= 1 to 5 do (* Nollställ frekvensräknare: *)
    frekv[betyg]:= 0;
  antal:= 0;
  summa:= 0;
  while not EOF(fil) do begin
    Read(fil, betyg);
    antal:= antal+1;
    frekv[betyg]:= frekv[betyg] + 1;
    summa:= summa+betyg;
  end; (* while *)
  Close(fil);
  WriteLn('Följande resultat erhöjls:');
  WriteLn(antal, ' betyg inlästa.');
```

```

  for betyg:= 1 to 5 do
    WriteLn(betyg, ':or = ', frekv[betyg]);
  WriteLn('Medelbetyg = ', summa/antal:1:1);
end.

```

```

program Ovn_9_3; (* Förutsätter data ligger i textfilen ARTIKEL.TXT *)
var
  textfil: Text;
  antal: Integer;
  pris, varde, total: Real;
  namn: string[20];
begin
  Assign(textfil, 'ARTIKEL.TXT'); Reset(textfil);
  total:= 0;
  WriteLn('Lagerlista'); WriteLn;
  WriteLn('Benämning           Antal           Pris           Värde');
  while not EOF(textfil) do begin
    ReadLn(textfil, namn);
    if not EOF(textfil) then begin
      ReadLn(textfil, antal, pris);
      varde:= antal*pris;
      total:= total + varde;
      WriteLn(namn,':20-Length(namn), antal:5,pris:10:2, varde:15:2);
    end;
  end;
  Close(f);
  WriteLn('Totalt':25, total:25:2);
end.

program Ovn_9_4;
var
  filnamn: string;
  textfil: Text;
  antalRader: Integer;
begin
  Write('Ange textfilens namn: '); ReadLn(filnamn);
  Assign(textfil, filnamn); Reset(textfil);
  antalRader:= 0;
  while not EOF(textfil) do begin
    ReadLn(textfil);
    antalRader:= antalRader + 1;
  end;
  Close(textfil);
  WriteLn('Filen innehåller ', antalRader, ' rader');
end.

program Ovn_9_5;
var
  filnamn, textrad: string;
  textfil: Text;
  antalTomrader: Integer;
begin
  Write('Ange textfilens namn: '); ReadLn(filnamn);
  Assign(textfil, filnamn); Reset(textfil);
  antalTomrader:= 0;
  while not EOF(textfil) do begin
    ReadLn(textfil, textrad);
    if textrad = '' then
      antalTomrader:= antalTomrader + 1;
  end;
  Close(textfil);
  WriteLn('Filen innehåller ', antalTomrader, ' tomrader');
end.

```

```

program Ovn_9_6a;
const
  filnamn = 'HELTAL.DAT';
var
  i, tal : Integer;
  textfil: Text;
begin
  Assign(textfil, filnamn); Rewrite(textfil);
  Randomize;
  for i:= 1 to 100 do begin
    tal:= Random(1000);
    Write(textfil, tal);
    if i mod 10 = 0 then
      WriteLn(textfil)
    else
      Write(textfil, ' ');
  end;
  Close(textfil);
end.

program Ovn_9_6b;
const
  filnamn = 'HELTAL.DAT';
var
  i, tal, max, min, radmin, radmax, radnr: Integer;
  textfil: Text;
  talsumma: LongInt;
begin
  Assign(textfil, filnamn); Reset(textfil);
  max:= 0; min:= 1000; radmax:= 0; radmin:= 1000;
  talsumma:= 0;
  radnr:= 0;
  WriteLn('Radnummer   Minsta   Största');
  while not EOF(textfil) do begin
    for i:= 1 to 10 do begin
      Read(textfil,tal);
      talsumma:= talsumma + tal;
      if tal > max then
        max:= tal
      else if tal < min then
        min:= tal;
      if tal > radmax then
        radmax:= tal
      else if tal < radmin then
        radmin:= tal;
    end;
    ReadLn(textfil);
    radnr:= radnr+1;
    WriteLn(radnr:4, radmin:13, radmax:10);
    radmax:= 0; radmin:= 1000;
  end;
  Close(textfil);
  WriteLn;
  WriteLn('Minsta talet i filen är ', min);
  WriteLn('Största talet i filen är ', max);
  WriteLn('Talsumman är ', talsumma);
end.

```

KAPITEL 10

```

program Ovn_10_1;
var
    frekvens          : array [1..6] of Integer;
    ogon, antalKast, kast: Integer;
    svar              : Char;
begin
    Randomize;
    repeat
        for ogon:= 1 to 6 do
            frekvens[ogon]:= 0;
            WriteLn('Kast med tärning. ');
            Write('Ange antalet kast: '); ReadLn(antalKast);
            for kast:= 1 to antalKast do begin
                ogon:= Random(6) + 1;
                frekvens[ogon]:= frekvens[ogon] + 1;
            end;
            WriteLn('Frekvenstabell. ');
            WriteLn;
            WriteLn('          Etta Tvåa Trea Fyra Femma Sexa ');
            Write('Frekvens (%) ');
            for ogon:= 1 to 6 do
                if ogon = 5 then
                    Write(frekvens[5]/kast*100:6:0)
                else
                    Write(frekvens[ogon]/kast*100:5:0);
                WriteLn;
            Write('Fler gånger? '); ReadLn(svar);
            until svar in ['n', 'N'];
    end.

```

```

program Ovn_10_2;
var
    vektor      : array [1..100] of Integer;
    k, n, temp: Integer;
begin
    Randomize;
    for k:= 1 to 100 do
        vektor[k]:= Random(1001);
    WriteLn('Slumptalen osorterade är: ');
    for k:= 1 to 100 do
        Write(vektor[k]:8);
    WriteLn;
    for k:= 1 to 99 do
        for n:= k+1 to 100 do
            if vektor[k] > vektor[n] then begin
                temp:= vektor[k];
                vektor[k]:= vektor[n];
                vektor[n]:= temp;
            end;
    WriteLn('Slumptalen sorterade i stigande ordning är: ');
    for k:= 1 to 100 do
        Write(vektor[k]:8);
    WriteLn;
end.

```

```

program Ovn_10_3;

type
  Heltalstyp = array[-1..255] of Char;

procedure LasTal(var x: Heltalstyp);
const
  okChars: set of Char = ['0'..'9', ' '];
var
  i, sifferant: Integer;
  talstr: string;
  ch: Char;
  ok, slut: Boolean;
begin
  repeat
    Write('Skriv ett stort heltal (. avslutar): ');
    ReadLn(talstr);
    ok:= true;
    i:= 1;
    while (i <= Length(talstr)) and ok do begin
      ok:= talstr[i] in okChars;
      i:= i + 1;
    end;
    slut:= talstr[1] = '.';
    if ok then begin
      sifferant:= 0;
      for i:= Length(talstr) downto 1 do
        if talstr[i] <> ' ' then begin
          x[sifferant]:= talstr[i];
          sifferant:= sifferant + 1;
        end;
        x[-1] := Chr(sifferant);
        ok:= sifferant > 0;
      end;
      if slut then
        x[-1]:= Chr(0) (* 0 siffror om slut *)
      else if not ok then
        WriteLn('Detta är inget tal!')
    until ok or slut;
  end;

procedure SkrivTal(var x: Heltalstyp);
var i: Integer;
begin
  for i:= Ord(x[-1])-1 downto 0 do
    Write(x[i]);
  end;

var x: Heltalstyp;

begin
  LasTal(x);
  while x[-1] > Chr(0) do begin (* Så länge som ett tal inlästs *)
    SkrivTal(x); WriteLn;
    LasTal(x);
  end;
end.

```

Övning 10.4:

```

procedure SkrivTal (var x: Heltalstyp);
var i: Integer;
begin
  for i:= Ord(x[-1]) -1 downto 0 do begin
    Write(x[i]);
    if i mod 3 = 0 then Write(' ');
  end;
end;

program Ovn_10_7;
var
  vektor: array [1..6] of Integer;
  k, antalNollor, summaNegativa,
  antalNegativa, summaPositiva, antalPositiva: Integer;
  flera: Boolean;
begin
  Randomize;
  for k:= 1 to 6 do
    vektor[k]:= Random(41) - 20;
  Write('Negativa tal: ');
  summaNegativa:= 0;
  antalNegativa:= 0;
  flera:= false;
  for k:= 1 to 6 do
    if vektor[k] < 0 then begin
      summaNegativa:= summaNegativa + vektor[k];
      antalNegativa:= antalNegativa + 1;
      if flera then
        Write(', ', vektor[k])
      else begin
        Write(vektor[k]);
        flera:= true;
      end;
    end;
  WriteLn(' Medelvärde: ', summaNegativa/antalNegativa:1:1);
  summaPositiva:= 0; antalPositiva:= 0;
  flera:= false;
  Write('Positiva tal: ');
  for k:= 1 to 6 do
    if vektor[k] > 0 then begin
      summaPositiva:= summaPositiva + vektor[k];
      antalPositiva:= antalPositiva + 1;
      if flera then
        Write(', ', vektor[k])
      else begin
        Write(vektor[k]);
        flera:= true;
      end;
    end;
  WriteLn(' Medelvärde: ', summaPositiva/antalPositiva:1:1);
  antalNollor:= 0;
  for k:= 1 to 6 do
    if vektor[k] = 0 then antalNollor:= antalNollor + 1;
  WriteLn('Talet noll förekommer ', antalNollor, ' gånger.');
```

```

end.

```

```

program Ovn_10_8;
var vektorA, vektorB: array [1..100] of Integer;
    nA, nB, A, B, indexA, indexB, k, n, temp: Integer;
begin
    Randomize;
    repeat
        Write('Ange önskat antal tal i vektor A (max 100): '); ReadLn(nA);
    until (nA >= 1) and (nA <= 100);
    repeat
        Write('Ange önskat antal tal i vektor B (max 100): '); ReadLn(nB);
    until (nB >= 1) and (nB <= 100);
    for k:= 1 to nA do vektorA[k]:= Random(120);
    for k:= 1 to nB do vektorB[k]:= Random(120);
    for k:= 1 to nA-1 do
        for n:= k+1 to nA do
            if vektorA[k] > vektorA[n] then begin
                temp:= vektorA[k]; vektorA[k]:= vektorA[n]; vektorA[n]:= temp;
            end;
        Write('Vektor A = ');
        for k:= 1 to nA do Write(vektorA[k]:8);
        WriteLn;
        for k:= 1 to nB-1 do
            for n:= k+1 to nB do
                if vektorB[k] > vektorB[n] then begin
                    temp:= vektorB[k]; vektorB[k]:= vektorB[n]; vektorB[n]:= temp;
                end;
            Write('Vektor B = ');
            for k:= 1 to nB do Write(vektorB[k]:8);
            WriteLn; WriteLn('Vektorerna A och B samsorterade:');
            indexA:= 1; indexB:= 1; A:= vektorA[indexA]; B:= vektorB[indexB];
            while (A <> maxInt) or (B <> maxInt) do
                if A < B then begin
                    Write(A:8);
                    if indexA < nA then begin
                        indexA:= indexA + 1; A:= vektorA[indexA];
                    end
                    else A:= maxInt;
                end
                else if A > B then begin
                    Write(B:8);
                    if indexB < nB then begin
                        indexB:= indexB + 1; B:= vektorB[indexB];
                    end
                    else B:= maxInt;
                end
                else begin (* Talen lika, skriv ut båda *)
                    Write(A:8, B:8);
                    if indexA < nA then begin
                        indexA:= indexA + 1; A:= vektorA[indexA];
                    end
                    else A:= maxInt;
                    if indexB < nB then begin
                        indexB:= indexB + 1; B:= vektorB[indexB];
                    end
                    else B:= maxInt;
                end;
            end.

```

```

program Ovn_10_10;
var
  deltagare: array [1..12] of string;
  nummer   : Integer;
begin
  WriteLn('Inläsning av deltagarlista. ');
  for nummer:= 1 to 12 do begin
    Write('Deltagare ', nummer:2, ' : '); ReadLn(deltagare[nummer]);
  end;
  Write('Nummer? '); ReadLn(nummer);
  while nummer <> 0 do begin
    if (1 <= nummer) and (nummer <= 12) then
      WriteLn(deltagare[nummer])
    else
      WriteLn('Finns ingen sådan deltagare!');
      Write('Nummer? '); ReadLn(nummer);
    end;
    WriteLn('Tack för idag!');
  end.

```

```

program Ovn_10_11;
type
  Arrrtyp = array [0..100] of Integer;

  procedure ShiftLeft(var a: Arrrtyp);
  var i: Integer;
  begin
    for i:= 99 downto 0 do
      a[i+1]:= a[i];
      a[0]:= 0;
    end;

  procedure SkrivUt(var a: Arrrtyp);
  var i: Integer;
  begin
    for i:= 0 to 100 do Write(a[i]:8);
    WriteLn;
  end;

var
  vektor   : Arrrtyp;
  i, n, temp: Integer;
begin
  Randomize;
  for i:= 0 to 100 do
    vektor[i]:= Random(100);
  SkrivUt(vektor);
  ShiftLeft(vektor);
  SkrivUt(vektor);
end.

```

```

program Ovn_10_12;
const maxelement = 6;
var heltal      : array[1..maxelement] of Integer;
    i, index, max: Integer;
begin
  for i:= 1 to maxelement do begin
    Write('Ange heltal nr ', i, ': '); ReadLn(heltal[i]);
  end;
  max:= heltal[1]; index:= 1;
  for i:= 2 to maxelement do
    if heltal[i] >= max then begin
      index:= i; max:= heltal[i];
    end;
  WriteLn('Största värdet är ', max, ' Index är ', index);
end.

```

```

program Ovn_10_13;
var poang      : array [1..6] of Real;
    domare      : Integer;
    min, max, summa: Real;
begin
  for domare:= 1 to 6 do begin
    Write('Poäng av domare ', domare, ': '); ReadLn(poang[domare]);
  end;
  summa:= poang[1]; min:= poang[1]; max:= poang[1];
  for domare:= 2 to 6 do begin
    if poang[domare] < min then
      min:= poang[domare]
    else if poang[domare] > max then
      max:= poang[domare];
    summa:= summa + poang[domare];
  end;
  summa:= summa - min - max;
  WriteLn('Medelpoäng = ', summa/4:1:2);
end.

```

```

program Ovn_10_14;
var
  vektor      : array [1..100] of Integer;
  antal, min, max, summa, nr: Integer;
begin
  repeat
    Write('Ange antalet heltal (max 100): '); ReadLn(antal);
  until (1 <= antal) and (antal <= 100);
  Write('Ange heltal 1: '); ReadLn(vektor[1]);
  min:= vektor[1]; max:= vektor[1]; summa:= vektor[1];
  for nr:= 2 to antal do begin
    Write('Ange heltal ', nr:3, ': '); ReadLn(vektor[nr]);
    if vektor[nr] < min then min:= vektor[nr]
    else if vektor[nr] > max then max:= vektor[nr];
    summa:= summa + vektor[nr];
  end;
  WriteLn('Summa: ', summa);
  WriteLn('Medelvärde: ', summa/antal:1:1);
  WriteLn('Minsta värde: ', min);
  WriteLn('Största värde: ', max);
end.

```

KAPITEL 11

```

program Ovn_11_1;

  procedure Rakna(n: Integer); (* OBS procedur, ej funktion *)
  begin
    if n > 0 then begin
      WriteLn(n);
      Rakna(n-1);
    end;
  end;

  var
    tal: Integer;
  begin
    Write('Ge ett heltal: '); ReadLn(tal);
    WriteLn('Nedräkning av talet ', tal, ' med hjälp av rekursion!');
    Rakna(tal);
  end.

```

Övning 11.2:

```

function Power(b: Real; n: Integer): Real;
begin
  if b = 0 then
    Power:= 0.0
  else if n < 0 then
    Power:= 1/Power(b, -n)
  else if n = 0 then
    Power:= 1
  else
    Power:= b*Power(b, n-1)
end (* Power *);

```

```

program Ovn_11_3; (* Med testprogram *)

  function Sum(n: Integer): Real; (* Kan bli ganska stort *)
  begin
    if n <= 1 then
      Sum:= 1
    else
      Sum:= Sum(n-1) + n;
    end;

  var
    n: Integer;
  begin
    Write('Ange n-värdet: '); ReadLn(n);
    WriteLn('Summan 1 + 2 + ... + ', n, ' = ', Sum(n):1:0);
  end.

```

```

program Ovn_11_4; (* Med testprogram *)

  function Sgd(a, b: Integer): Integer;
  begin
    if a = b then
      Sgd:= a
    else if a > b then
      Sgd:= Sgd (a-b, b)
    else
      Sgd:= Sgd(a, b-a);
  end;

  var
    t1, t2: Integer;
  begin
    Write('Ange första heltalet: '); ReadLn(t1);
    Write('Ange andra heltalet : '); ReadLn(t2);
    WriteLn('Största delare till ', t1, ' och ', t2, ' = ', Sgd(t1,t2));
  end.

```

```

program Ovn_11_5; (* Med testprogram *)

  function SiffSum(n: LongInt): Integer;
  begin
    if n <= 0 then
      SiffSum:= 0
    else
      SiffSum:= SiffSum(n div 10) + n mod 10;
  end;

  var
    n: LongInt;
  begin
    Write('Ange n-värdet: '); ReadLn(n);
    WriteLn('Siffersumman = ', SiffSum(Abs(n)));
  end.

```

```

program Ovn_11_6; (* Med testprogram *)

  function Palin(s: string): Boolean;
  begin
    if (Length(s) <= 1) then
      Palin:= true
    else if s[1] <> s[Length(s)] then
      Palin:= false
    else
      Palin:= Palin(Copy(s, 2, Length(s)-2))
  end; (* Palin *)

  var
    s: string;
  begin
    Write('Skriv en sträng: '); ReadLn(s);
    if not Palin(s) then Write('ICKE ');
    WriteLn('palindrom');
  end.

```

```
program Ovn_11_7; (* Med testprogram *)

type
  Str80 = string[80];

function DecTillBin(n: LongInt): Str80;
begin
  if n = 0 then
    DecTillBin:= '0'
  else if n = 1 then
    DecTillBin:= '1'
  else
    DecTillBin:= DecTillBin(n div 2) + Chr(Ord('0')+n mod 2);
end;

var
  dectal: LongInt;
begin
  Write('Ange ett positivt heltal (0 avslutar): ');
  ReadLn(dectal);
  while dectal <> 0 do begin
    Write('Svar: ', DecTillBin(dectal));
    WriteLn;
    Write('Ange ett positivt heltal (0 avslutar): ');
    ReadLn(dectal);
  end;
end.
```

KAPITEL 12*Övning 12.1:*

```

unit MittNamn;

interface (* -----*)
const namnet = 'Sven Svensson';

procedure Namn;

implementation (* -----*)
uses CRT;

    procedure Namn;
    begin
        ClrScr;
        GotoXY(40 - Length(namnet) div 2, 12);
        Write(namnet);
    end;

end.

program Ovn_12_1;
uses MittNamn;
begin
    Namn;
end.

unit Cirkel;

interface (* -----*)

function Area(radie: Real): Real;
function Omkrets(radie: Real): Real;

implementation (* -----*)

    function Area(radie: Real): Real;
    begin
        Area:= Pi*radie*radie;
    end;

    function Omkrets(radie: Real): Real;
    begin
        Omkrets:= 2*Pi*radie;
    end;
end.

program Ovn_12_2;
uses Cirkel;
var radie: Real;
begin
    Write('Ange cirkelns radie: '); ReadLn(radie);
    WriteLn('Arean blir ', Area(radie):1:2);
    WriteLn('Omkretsen blir ', Omkrets(radie):1:2);
end.

```

```
unit Variab;  
  
interface  
  
var  
    tall, tal2, sum: Integer;  
  
implementation  
  
end.
```

Övning 12.5:

```
unit Ber;  
  
interface  
(* uses Variab; Behövs ej här om tall och tal2 skickas som  
   parametrar till funktionen, som nedan *)  
  
function Summa(tall, tal2: Integer): Integer;  
  
implementation  
  
function Summa(tall, tal2: Integer): Integer;  
begin  
    Summa:= tall+tal2;  
end;  
  
end.  
  
program Ovn_12_5;  
uses Variab, Ber;  
begin  
    WriteLn('Detta är ett program som summerar två tal');  
    WriteLn;  
    Write('Ange det första talet: '); ReadLn(tall);  
    Write('Ange det andra talet: '); ReadLn(tal2);  
    sum:= Summa(tall, tal2);  
    WriteLn('Summan blir ', sum);  
end.
```

KAPITEL 13

Övning 13.1: Se type-deklarationen i nästa uppgift

```

program Ovn_13_2;
uses CRT;

const maxantal = 100;
type
  Str8 = string[8];
  Str18 = string[18];
  Artikel = record
    artnr      : Str8;
    artnamn    : Str18;
    lagerantal: Integer;
    styckpris  : Real;
end;
Lagertyp = array[1..maxantal] of Artikel;

procedure Inmata(var lager: Lagertyp; var antal: Integer);
var temp: Str8;
begin
  ClrScr;
  WriteLn('INMATNING AV ARTIKLAR'); WriteLn;
  Write('Ange artikelnummer (avbryt med Enter): '); ReadLn(temp);
  while (temp <> '') and (antal < maxantal) do begin
    antal:= antal + 1;
    with lager[antal] do begin
      artnr:= temp;
      Write('Ange artikelnamn: '); ReadLn(artnamn);
      Write('Ange antal i lager: '); ReadLn(lagerantal);
      Write('Ange styckepris: '); ReadLn(styckpris);
    end; (* with *)
    Write('Ange artikelnummer (avbryt med Enter): '); ReadLn(temp);
  end; (* while *)
end;

procedure Lagerlista(var lager: Lagertyp; antal: Integer);
var i: Integer;
begin
  ClrScr;
  WriteLn('LAGERLISTA'); WriteLn; WriteLn;
  WriteLn('Artikelnr  Artikelnamn          Antal i lager      Pris');
  WriteLn('=====');
  WriteLn;
  for i:= 1 to antal do begin
    with lager[i] do begin
      WriteLn(artnr:9, ':2, artnamn, ':18-Length(artnamn),
        lagerantal:11, styckpris:15:2);
    end; (* with *)
    if i mod 16 = 0 then begin
      WriteLn;
      WriteLn('Tryck Enter för forts':60); ReadLn;
      WriteLn;
    end; (* if *)
  end; (* for *)
end;

```

```

var lager: Lagertyp;
    antal: Integer;
    ch   : Char;
begin
    antal:= 0;
    Inmata(lager, antal);
    Write('Vill du ha en lagerlista (J/N): '); ReadLn(ch);
    if ch in ['J', 'j'] then Lagerlista(lager, antal);
end.

program Ovn_13_3;
type
    Str10 = string[10];
    Str25 = string[25];
    Personpost = record
        fornamn,
        efternamn: Str25;
        tfn: Str10;
    end;
    Klassregister = array[1..30] of Personpost;

procedure Sok(var elev: Klassregister; antal: Integer);
var namn: Str25;
    i: Integer;
    funnen: Boolean;
begin
    Write('Ange elevens efternamn: '); ReadLn(namn);
    funnen:= false;
    for i:= 1 to antal do begin
        if namn = elev[i].efternamn then begin
            funnen:= true;
            with elev[i] do begin
                WriteLn('Förnamn : ', fornamn);
                WriteLn('Efternamn: ', efternamn);
                WriteLn('Telefon : ', tfn);
            end;
            WriteLn;
        end;
    end;
    if not funnen then WriteLn('ELEV MED DETTA EFTERNAMN FINNS EJ!');
end; (* Sok *)

var elev : Klassregister;
    antal: Integer;
    namn : Str25;
begin
    antal:= 0;
    Write('Ange förnamn (avsluta med "tomt" namn): '); ReadLn(namn);
    while (namn <> '') and (antal < 30) do begin
        antal:= antal + 1; elev[antal].fornamn:= namn;
        Write('Ange efternamn: '); ReadLn(elev[antal].efternamn);
        Write('Ange tfnr: '); ReadLn(elev[antal].tfn);
        Write('Ange förnamn (avsluta med "tomt" namn): '); ReadLn(namn);
    end;
    Sok(elev, antal);
end.

```

```

program Ovn_13_4;

const maxantal = 30;
type
  Artikelposttyp = record
    artikelnummer: string[10];
    artikelnamn  : string[20];
    artikelantal : Integer;
    styckpris    : Real;
  end;
  Artikelregtyp = array[1..maxantal] of Artikelposttyp;

procedure Inmatning(var artreg: Artikelregtyp);
var antal: Integer;
begin
  for antal:= 1 to maxantal do
    with artreg[antal] do begin
      Write('Ange artikelnummer: '); ReadLn(artikelnummer);
      Write('Ange artikelnamn: '); ReadLn(artikelnamn);
      Write('Ange artikelantal: '); ReadLn(artikelantal);
      Write('Ange styckpris: '); ReadLn(styckpris);
    end;
  end;

procedure Utskrift(artreg: Artikelregtyp);
var antal: Integer;
begin
  WriteLn('Artikelregister'); WriteLn; WriteLn;
  for antal:= 1 to maxantal do begin
    with artreg[antal] do begin (* fullfölj *)
      end;
    if antal mod 15 = 0 then ReadLn;
  end;
end;

procedure Andring(var artreg: Artikelregtyp);
var
  n, andring: Integer;
  soknr      : string[10];
begin
  Write('Ange sökt artikelnummer/<Enter>: '); ReadLn(soknr);
  while soknr <> '' do begin
    n:= 0;
    repeat
      n:= n + 1;
    until (soknr = artreg[n].artikelnummer) or (n = maxantal);
    if soknr = artreg[n].artikelnummer then begin
      WriteLn('Nuvarande artikelantal: ', artreg[n].artikelantal);
      Write('Ange förändring: '); ReadLn(andring);
      artreg[n].artikelantal:=artreg[n].artikelantal+andring;
    end
    else
      WriteLn(soknr, ' finns inte i registret!');
      Write('Ange sökt artikelnummer/<Enter>: '); ReadLn(soknr);
    end;
  end;
end;

```

```

procedure Sokning(artreg: Artikelregtyp);
var
    n      : Integer;
    soknr: string[10];
begin
    Write('Ange sökt artikelnummer/<Enter>: '); ReadLn(soknr);
    while soknr <> '' do begin
        n:= 0;
        repeat
            n:= n + 1;
        until (soknr = artreg[n].artikelnummer) or (n = maxantal);
        if soknr = artreg[n].artikelnummer then begin
            WriteLn('Artikelnummer Artikelnamn Antal Styckpris');
            with artreg[n] do begin
                Write(artikelnummer, ':15-Length(artikelnummer), artikelnamn);
                WriteLn(artikelantal:25-Length(artikelnamn), styckpris:11:2);
            end;
        end
        else
            WriteLn(soknr, ' finns inte i registret!');
            Write('Ange sökt artikelnummer/<Enter>: '); ReadLn(soknr);
        end;
    end;

var
    artreg: Artikelregtyp;
    alternativ: Char;
begin
    repeat
        WriteLn('1. Inläsning av 30 artiklar. ');
        WriteLn('2. Utskrift av samtliga artiklar. ');
        WriteLn('3. Ändring av antal i lager. ');
        WriteLn('4. Sökning på visst artikelnummer. ');
        WriteLn;
        Write('Ange alternativ/(0): '); ReadLn(alternativ);
        case alternativ of
            '1': Inmatning(artreg);
            '2': Utskrift(artreg);
            '3': Andring(artreg);
            '4': Sokning(artreg);
        end;
    until alternativ = '0';
end.

program Ovn_13_5;

type
    Posttyp = record
        ben  : string[30];
        antal: Integer;
        pris : Real;
    end;
    Lagertyp = array [1..100] of Posttyp;

```

```

procedure LasIn(var lager: Lagertyp; var antArt: Byte);
var benamn: string[30];
begin
  Write('Ange artikelbenämning/<Enter>: '); ReadLn(benamn);
  while (benamn <> '') and (antArt < 100) do begin
    antArt:= antArt + 1;
    with lager[antArt] do begin
      ben:= benamn;
      Write('Ange antal: '); ReadLn(antal);
      Write('Ange pris: '); ReadLn(pris);
    end;
    Write('Ange artikelbenämning/<Enter>: '); ReadLn(benamn);
  end;
end;

procedure PrisAndring(var lager: Lagertyp; antArt: Byte);
var i      : Integer;
    totalt, procent: Real;
begin
  Write('Ange hur många procent prishöjning: '); ReadLn(procent);
  WriteLn('Artikelbenämning', 'Pris':24); WriteLn;
  totalt:= 0;
  for i:= 1 to antArt do
    with lager[i] do begin
      pris:= (1 + procent/100)*pris;
      WriteLn(ben, pris:40-Length(ben):2);
      totalt:= totalt + antal*pris;
    end;
  WriteLn('Totala lagervärdet = ', totalt:19:2);
end;

var lager : Lagertyp;
    antArt: Byte;
begin
  antArt:= 0;
  LasIn(lager, antArt);
  PrisAndring(lager, antArt);
end.

```

Övning 13.6:

Uppgiften är klumpigt formulerad i läroboken. Det är t ex minnesslöseri att använda datatypen **string** (rymmer upp till 255 tecken) för namn och boktitlar. Det är också en onödig och konstlad begränsning att förutsätta att "listan är full". Denna sista inskränkning har nedan undanröjts genom att komplettera procedurerna med en antalsparameter (alternativt kan en global variabel tillgripas).

```

const max = 50;
type
  Forfattartyp = record
    fornamn, efternamn: string;
  end;
  Boktyp = record
    titel: string;
    forfattare: Forfattartyp;
    pris: Real;
  end;
  Listtyp = array[1..max] of Boktyp;

```

```

procedure Billigast(var lista: Listtyp; bokantal: Integer);
(* Skriver ut första påträffade bok med lägsta pris *)
var
  i, minix: Integer;
  minpris: Real;
begin
  minpris:= lista[1].pris;
  minix:= 1;
  for i:= 2 to bokantal do begin
    if lista[i].pris < minpris then begin
      minpris:= lista[i].pris;
      minix:= i;
    end;
  end;
  WriteLn('Billigast är "', lista[minix].titel,
    '". Den kostar ', lista[minix].pris:1:2);
end;

function Antal(fnamn, enamn: string;
  var lista: Listtyp; bokantal: Integer): Byte;
var
  i: Integer;
  n: Byte;
begin
  n:= 0;
  for i:= 1 to bokantal do begin
    with lista[i].forfattare do
      if (fornamn = fnamn) and (efternamn = enamn) then
        n:= n+1;
    end;
  Antal:= n;
end;

program Ovn_13_7;
const
  max = 5;
type
  Elposttyp = record
    m2, kWh: Integer;
  end;
  Lagenhetstyp = array [1..max] of Elposttyp;

procedure Inmatning(var lagenhet: Lagenhetstyp);
var
  antal: Integer;
begin
  WriteLn('Bostadsstorlek i kvadratmeter och elförbrukning i kWh');
  for antal:= 1 to max do begin
    WriteLn('lägenhet nr ', antal);
    Write('storlek: '); ReadLn(lagenhet[antal].m2);
    Write('förbrukning: '); ReadLn(lagenhet[antal].kWh);
    WriteLn;
  end;
end;

```

```

procedure Medelforbrukning(lagenhet: Lagenhetstyp);
var
  minYta, maxYta, lagenhetsAntal, antal: Integer;
  forbrukning                : LongInt;
  svar                       : Char;
begin
  repeat
    WriteLn('Medelförbrukning i visst storleksintervall');
    WriteLn('Ange intervallets gränser. ');
    Write('Minsta yta: '); ReadLn(minYta);
    Write('Största yta: '); ReadLn(maxYta);
    lagenhetsAntal:= 0;
    forbrukning:= 0;
    for antal:= 1 to max do
      if (lagenhet[antal].m2 >= minYta) and
        (lagenhet[antal].m2 <= maxYta) then begin
        lagenhetsAntal:= lagenhetsAntal + 1;
        forbrukning:= forbrukning + lagenhet[antal].kWh;
      end;
    Write('Intervall: ', minYta, '-', maxYta);
    Write('  Antal lägenheter: ', lagenhetsAntal);
    Write('  Medelförbrukning: ');
    if lagenhetsAntal = 0 then
      WriteLn('-')
    else
      WriteLn(forbrukning div lagenhetsAntal);
      Write('Nytt intervall? '); ReadLn(svar);
    until svar in ['n', 'N'];
end;

var
  lagenhet: Lagenhetstyp;
begin
  Inmatning(lagenhet);
  Medelforbrukning(lagenhet);
end.

program Ovn_13_8;
type
  ReellerKtyp = (reell, komplex);
  Losningstyp = record
    case losning: ReellerKtyp of
      reell: (x1, x2: Real);
      komplex: (re, im: Real);
    end;

var
  x: Losningstyp;
  q, r, p, d: Real;
begin
  WriteLn('Programmet beräknar rötterna till andragradsekvationen:');
  WriteLn('          px2 + qx + r = 0');
  Write('Ange värde på p: '); ReadLn(p);
  Write('Ange värde på q: '); ReadLn(q);
  Write('Ange värde på r: '); ReadLn(r);

```

```

if p = 0 then
  WriteLn('Detta är ingen andragradsekvation')
else begin
  q:= q/p; r:= r/p;
  d:= q*q - 4*r;
  if d>=0 then begin
    x.losning:= reell;
    if d=0 then begin (* Dubbelrot *)
      x.x1:= -q/2; x.x2:= x.x1;
    end
    else begin
      if q<0 then
        x.x1:= (-q + Sqrt(d))/2
      else
        x.x1:= (-q - Sqrt(d))/2;
        x.x2:= r/x.x1;
      end;
    end
  else begin
    x.losning:= komplex;
    x.re:= -q/2; x.im:= Sqrt(-d)/2;
  end;
  if x.losning = reell then begin
    WriteLn('Ekvationen har reella rötter:');
    WriteLn('x1 = ', x.x1:1:2);
    WriteLn('x2 = ', x.x2:1:2);
  end
  else begin
    WriteLn('Ekvationen har komplexa rötter:');
    WriteLn('x1 = ', x.re:1:2, ' + ', x.im:1:2, 'i');
    WriteLn('x2 = ', x.re:1:2, ' - ', x.im:1:2, 'i');
  end;
  end;
end.

```

KAPITEL 14

```
program Ovn_14_2;
type
  Talfil = file of Byte;
var
  fil: Talfil;

  procedure NyFil;
  begin
    Assign(fil, 'TAL.DAT'); Rewrite(fil);
  end;

  procedure LagraTal;
  var
    i, antal: Integer;
    tal: Byte;
  begin
    Randomize;
    antal:= Random(31)+20; (* 20 -50 st *)
    for i:= 1 to antal do begin
      tal:= Random(101);
      Write(fil, tal);
    end;
    Close(fil);
  end;

  procedure LasTal;
  var
    tal: Byte;
  begin
    Reset(fil);
    while not EOF(fil) do begin
      Read(fil, tal);
      Write(tal:4);
    end;
    WriteLn;
  end;

begin
  NyFil;
  LagraTal;
  LasTal;
  WriteLn;
  WriteLn('Det är ', FileSize(fil), ' tal lagrade på filen');
  Close(fil);
end.
```

```

program Ovn_14_6;
type
  Str8 = string[8];
  Str40 = string[40];
  Skolkare = record
    namn : Str40;
    klass,
    tel : Str8;
    timmar: Integer;
  end;

var skolkfil: file of Skolkare;

procedure SkrivSkolklista;

  procedure Skollista(klass: Str8; franvaro: Integer);
  var elev: Skolkare;
  begin
    Write('Elever i ', klass);
    WriteLn(' med frånvaro större än ', franvaro, ' timmar. ');
    Assign(skolkfil, 'SKOLK.DAT'); Reset(skolkfil);
    while not EOF(skolkfil) do begin
      Read(skolkfil, elev);
      if (elev.klass = klass) and (elev.timmar > franvaro) then
        WriteLn(elev.namn, elev.timmar:40-Length(elev.namn));
      end;
      Close(skolkfil);
    end;

var klass : Str8;
    franvaro: Integer;
begin
  Write('Ange klass: '); ReadLn(klass);
  while klass <> '' do begin
    Write('Ange frånvarotimmar: '); ReadLn(franvaro);
    Skollista (klass, franvaro);
    WriteLn;
    Write('Ange klass: '); ReadLn(klass);
  end;
end;

procedure UppdateraFranvaro;

  procedure Uppdatera(namn: Str40; klass: Str8; franvarotal: Integer);
  var elev: Skolkare;
  begin
    Assign(skolkfil, 'SKOLK.DAT'); Reset(skolkfil);
    while not EOF(skolkfil) do begin
      Read(skolkfil, elev);
      if (elev.namn = namn) and (elev.klass = klass) then begin
        postnr:= FilePos(skolkfil);
        elev.timmar:= elev.timmar + franvarotal;
        Seek(skolkfil, postnr-1); Write(skolkfil, elev);
      end
      else WriteLn('Eleven finns ej med i registret!');
      end;
      Close(skolkfil);
    end;
  end;

```

```

var
  namn      : Str40;
  klass     : Str8;
  franvaro: Integer;
begin
  Write('Ange namn: '); ReadLn(namn);
  while namn <> '' do begin
    Write('Ange klass: '); ReadLn(klass);
    Write('Ange frånvarotimmar: '); ReadLn(franvaro);
    Uppdatera(namn, klass, franvaro);
    WriteLn;
    Write('Ange namn: '); ReadLn(namn);
  end;
end;

var alternativ: Char;
begin
  repeat
    WriteLn('1. Skriv Skolklista');
    WriteLn('2. Uppdatera Frånvaro');
    WriteLn('0. Avsluta');
    WriteLn;
    Write('Ange alternativ: '); ReadLn(alternativ);
    case alternativ of
      '1': SkrivSkolklista;
      '2': UppdateraFranvaro;
    end;
  until alternativ = '0';
end.

```

Xtra: Program för att skapa datafil

```

program SkapaSkolkfil;
type
  Str8  = string[8];
  Str40 = string[40];
  Skolkare = record
    namn  : Str40;
    klass,
    tel   : Str8;
    timmar: Integer;
  end;

var skolkpost: Skolkare;
    skolkfil : file of Skolkare;
begin
  Assign(skolkfil, 'SKOLK.DAT'); Rewrite(skolkfil);
  Write('Ange namn: '); ReadLn(skolkpost.namn);
  while skolkpost.namn <> '' do begin
    Write('Ange klass: '); ReadLn(skolkpost.klass);
    Write('Ange telnr: '); ReadLn(skolkpost.tel);
    Write('Ange timmar: '); ReadLn(skolkpost.timmar);
    Write(skolkfil, skolkpost);
    Write('Ange namn: '); ReadLn(skolkpost.namn);
  end;
  Close(skolkfil);
end.

```

```

program Ovn_14_7;
(* Svaret på övningen omfattar egentligen bara procedurerna Lagervarde
och FinnBil. Huvudprogram liksom ett program för att skapa datafil
(se nästa sida) har tillagts för att kunna testa det hela *)
const
  filnamn = 'BILFIL.DAT';

type
  Str25 = string[25];
  Biltyp = record
    marke,
    farg: Str25;
    ar: Integer;
    pris: Real;
  end;
  Bilfilstyp = file of Biltyp;

var
  bilfil: Bilfilstyp;

procedure Lagervarde;
var
  bil : Biltyp;
  summa: Real;
begin
  Reset(bilfil); (* Till början av filen *)
  summa:= 0;
  while not EOF(bilfil) do begin
    Read(bilfil, bil);
    summa:= summa + bil.pris;
  end;
  Close(bilfil);
  WriteLn('Lagervärde ', summa:1:2, ' kr');
end;

procedure FinnBil(bilmarke: Str25; ar: Integer);
var
  bil : Biltyp;
  funnen: Boolean;
begin
  funnen:= false;
  Reset(bilfil); (* Till början av filen *)
  WriteLn(bilmarke, ' av årsmodell ', ar, ':');
  WriteLn('Färg                Pris');
  while not EOF(bilfil) do begin
    Read(bilfil, bil);
    if (bilmarke = bil.marke) and (bil.ar = ar) then begin
      funnen:= true;
      WriteLn(bil.farg, ': 30-Length(bil.farg), bil.pris:1:0);
    end;
  end;
  Close(bilfil);
  if not funnen then
    WriteLn('Ingen bil av detta slag finns i lager!');
end;

```

```

var
    marke: Str25;
    ar: Integer;
begin
    Assign(bilfil, filnamn);
    Lagervarde;
    Write('Ange sökt bilmärke: '); ReadLn(marke);
    Write('Ange årsmodell: '); ReadLn(ar);
    FinnBil(marke, ar);
end.

program Ovn_14_X_7;
(* Program för att skapa bildatafil till övning 14.7 *)
const
    filnamn = 'BILFIL.DAT';
type
    Str25 = string[25];
    Biltyp = record
        marke,
        farg: Str25;
        ar: Integer;
        pris: Real;
    end;
    Bilfilstyp = file of Biltyp;
var
    bilfil: Bilfilstyp;
    bil: Biltyp;
begin
    Assign(bilfil, filnamn);
    Rewrite(bilfil);
    Write('Bilmärke (Sluta = <Enter>): '); ReadLn(bil.marke);
    while bil.marke <> '' do begin
        Write('Färg: '); ReadLn(bil.farg);
        Write('Årsmodell: '); ReadLn(bil.ar);
        Write('Pris: '); ReadLn(bil.pris);
        Write(bilfil, bil);
        Write('Bilmärke (Sluta = <Enter>): '); ReadLn(bil.marke);
    end;
    Close(bilfil);
end.

```

```

program Ovn_14_9;
type
  Talfil = file of Integer; (* Alternativt Word *)

procedure SkapaTalfil(var fil: Talfil);
(* Skapar en fil av det slag som resten av programmet behöver. *)
var
  i, antal, tal: Integer;
begin
  Rewrite(fil);
  Randomize;
  antal:= Random(981)+20; (* 20-1000 st *)
  for i:= 1 to antal do begin
    tal:= Random(1000) + 1; (* Talvärdet 1-1000 *)
    Write(fil, tal);
  end;
  Close(fil);
end;

procedure LasTal(var tal: Integer);
begin
  Write('Ange ett heltal (0 avslutar programmet): '); ReadLn(tal);
end;

procedure SokTal(var fil: Talfil; tal: Integer);
var
  x: Integer;
  plats: LongInt;
  funnet: Boolean;
begin
  Reset(fil); (* Till början av filen *)
  funnet:= false;
  while not EOF(fil) and not funnet do begin
    Read(fil, x);
    if x = tal then begin
      funnet:= true; plats:= FilePos(fil);
    end;
  end;
  if funnet then
    WriteLn('Talet ', tal, ' finns på plats ', plats, ' i filen.')
  else
    WriteLn('Talet ', tal, ' finns inte i filen');
end;

var
  fil: Talfil;
  tal: Integer;
begin
  Assign(fil, 'TAL.DAT');
  SkapaTalfil(fil); (* Behövs bara för att skapa datafil *)
  LasTal(tal);
  while tal <> 0 do begin
    SokTal(fil, tal);
    LasTal(tal);
  end;
  WriteLn('Tack för idag!');
  Close(fil);
end.

```

```

program Ovn_14_11;
type
  Frekvenstabell = array[1..5] of Integer;

procedure LasIn(var frekv: Frekvenstabell; var elevantal: Integer);
var
  fil : file of Integer;
  betyg: Integer;
begin
  Assign(fil, 'CPMA92.DAT'); Reset(fil);
  for betyg:= 1 to 5 do
    frekv[betyg]:= 0;
  elevantal:= 0;
  while not EOF(fil) do begin
    Read(fil, betyg);
    elevantal:= elevantal+1;
    frekv[betyg]:= frekv[betyg]+1;
  end;
  Close(fil);
end;

procedure SkrivUt(var frekv: Frekvenstabell; elevantal: Integer);
var
  betygssumma: Real;
  betyg: Integer;
begin
  betygssumma:= 0;
  for betyg:= 1 to 5 do
    betygssumma:= betygssumma + frekv[betyg]*betyg;
  WriteLn('Resultat av centralt prov i matematik 1992');
  WriteLn;
  WriteLn('Antal elever: ', elevantal);
  WriteLn('Betygsgenomsnitt: ', betygssumma/elevantal:1:1);
  WriteLn;
  WriteLn('Betygsfördelning:');
  WriteLn;
  Write('Betyg      ');
  for betyg:= 1 to 5 do
    Write(betyg:4);
  WriteLn;
  Write('Antal      ');
  for betyg:= 1 to 5 do
    Write(frekv[betyg]:4);
  WriteLn;
end;

var
  frekvens: Frekvenstabell;
  elevantal: Integer;
begin
  LasIn(frekvens, elevantal);
  SkrivUt(frekvens, elevantal);
end.

```

```
program Ovn_14_X_11; (* Skapar betygsfil till uppgift 14.11 *)  
var  
    fil: file of Integer;  
    betyg, elevantal, i, slumptal, offset: Integer;  
begin  
    Assign(fil, 'CPMA92.DAT'); Rewrite(fil);  
    Randomize;  
    elevantal:= Random(100) + 100; offset:= Random(18);  
    for i:= 1 to elevantal do begin  
        slumptal:= Random(91) + offset;  
        if slumptal < 8 then  
            betyg:= 1  
        else if slumptal < 31 then  
            betyg:= 2  
        else if slumptal < 69 then  
            betyg:= 3  
        else if slumptal < 93 then  
            betyg:= 4  
        else  
            betyg:= 5;  
        Write(fil, betyg);  
    end;  
    Close(fil);  
end.
```

```

program Ovn_14_12;
(*
Har ersatt string (som lagrar upp till 255 tecken långa strängar)
med kortare strängar (Str15 och Str40). Dessutom antas ej att filen
är öppen, vilket sägs i läroboken (man bör inte anta detta, eftersom
filpekaren kan stå var som helst i filen om man inte "resettar" den
*)

type
  Str15 = string[15];
  Str40 = string[40];
  Persdatatyp = record
    land   : Str15;
    alder,
    vikt   : Byte;
    persrek: Real;
end;

  Deltagtyp = record
    startnr: Byte;
    namn   : Str40;
    gren   : Str40;
    tid    : Real;
    persdata: Persdatatyp;
end;

  Filtyp = file of Deltagtyp;

var
  resfil: Filtyp; (* Global enligt boken. Borde skickas som
                  parameter till underprogrammen i stället *)

procedure SkrivUtPerson;
var
  nummer: Byte;
  delt: Deltagtyp;
  funnen: Boolean;
begin
  Write('Ange deltagarnummer: '); ReadLn(nummer);
  while nummer > 0 do begin
    funnen:= false;
    Reset(resfil);
    while not EOF(resfil) and not funnen do begin
      Read(resfil, delt);
      if delt.startnr = nummer then begin
        funnen:= true;
        WriteLn('Namn: ', delt.namn);
        WriteLn('Gren: ', delt.gren);
        WriteLn('Personligt rekord: ', delt.persdata.persrek:1:2);
      end;
    end;
    if not funnen then
      WriteLn('Finns ingen med detta nummer');
    Write('Ange nytt deltagarnummer (0 avslutar): ');
    ReadLn(nummer);
  end;
  Close(resfil);
end;

```

```

procedure Prislستا(gren: Str15);
const max = 50;
type
  Prispost = record
    namn: Str40;
    gren: Str40;
    tid : Real;
    land: Str15;
  end;
  Prisarray = array[1..max] of Prispost;

procedure Sortera(var a: Prisarray; antal: Integer);
var
  i, j, minIndex: Integer;
  temp : Prispost;
  minst: Real;
begin
  for i:= 1 to antal-1 do begin
    minIndex:= i; minst:= a[i].tid;
    for j:= i+1 to antal do
      if minst > a[j].tid then begin
        minIndex:= j; minst:= a[j].tid;
      end; (* if *)
      temp:= a[i];          (* Byt *)
      a[i]:= a[minIndex];
      a[minIndex]:= temp;
    end;
  end; (* Sortera *)

var
  delt: Deltagtyp;
  antal, i: Integer;
  a: Prisarray;
begin
  Reset(resfil); antal:= 0;
  while not EOF(resfil) and (antal < max) do begin
    Read(resfil, delt);
    if delt.gren = gren then begin
      antal:= antal + 1;
      a[antal].namn:= delt.namn;
      a[antal].gren:= delt.gren;
      a[antal].tid := delt.tid;
      a[antal].land:= delt.persdata.land;
    end;
  end;
  Close(resfil);
  if antal > 1 then Sortera(a, antal);
  if antal > 0 then begin
    WriteLn('PRISLISTA ', gren); WriteLn;
    WriteLn('Guld   ', a[1].namn, ' ', a[1].tid:1:2, ' ', a[1].land);
  end;
  if antal > 1 then
    WriteLn('Silver ', a[2].namn, ' ', a[2].tid:1:2, ' ', a[2].land);
  if antal > 2 then
    WriteLn('Brons  ', a[3].namn, ' ', a[3].tid:1:2, ' ', a[3].land);
  for i:= 4 to antal do
    WriteLn(i, '    ', a[i].namn, ' ', a[i].tid:1:2, ' ', a[i].land);
end;

```

```

procedure Landlista(land: Str15);
var
    deltag: Deltagtyp;
begin
    Reset(resfil);
    WriteLn('Startnr Namn                               Gren');
    while not EOF(resfil) do begin
        Read(resfil, deltag);
        if deltag.persdata.land = land then begin
            with deltag do
                WriteLn(startnr:4, ':4, namn, ':30-Length(namn), gren);
            end;
        end;
    Close(resfil);
end;

procedure SkapaFil;
var
    deltag: Deltagtyp;
begin
    Rewrite(resfil);
    with deltag do begin
        Write('Ange gren (avsluta med tom sträng: '); ReadLn(gren);
        while gren <> '' do begin
            Write('Ange namn (avsluta med tom sträng): '); ReadLn(namn);
            while namn <> '' do begin
                Write('Startnummer: '); ReadLn(startnr);
                Write('Tid: '); ReadLn(tid);
                Write('Land: '); ReadLn(persdata.land);
                Write('Ålder: '); ReadLn(persdata.alder);
                Write('Vikt: '); ReadLn(persdata.vikt);
                Write('Personligt rekord: '); ReadLn(persdata.persrek);
                Write(resfil, deltag);
                Write('Ange namn (avsluta med tom sträng): '); ReadLn(namn);
            end;
            Write('Ange gren (avsluta med tom sträng: '); ReadLn(gren);
        end;
    end;
    Close(resfil);
end;

var
    temp: Str15;
begin (* Test *)
    Assign(resfil, 'RES.DAT');
    (* SkapaFil; *) (* Avlägsna kommentartecken för att skapa filen *)

    SkrivUtPerson;

    Write('Ange land: '); ReadLn(temp);
    Landlista(temp);

    Write('Ange gren: '); ReadLn(temp);
    Prislista(temp);
end.

```

KAPITEL 15

```

program Ovn_15_1;
var
  ip: ^Integer;
begin
  New(ip);
  Write('Skriv ett heltal: '); ReadLn(ip^);
  WriteLn('Heltalet är ', ip^);
  Dispose(ip);
end.

program Ovn_15_3;
var
  tal1, tal2, sum: ^Integer;
begin
  New(tal1);
  New(tal2);
  New(sum);
  Write('Skriv ett heltal: '); ReadLn(tal1^);
  Write('Skriv ett heltal till: '); ReadLn(tal2^);
  sum^:= tal1^ + tal2^;
  WriteLn('Summan är ', sum^);
  Dispose(tal1);
  Dispose(tal2);
  Dispose(sum);
end.

program Ovn_15_4;
type
  Talvektortyp = array [1..10] of Integer;
var
  v1, v2: ^Talvektortyp;
  i: Integer;
  sum: LongInt;
begin
  New(v1);
  New(v2);
  WriteLn('Första vektorn');
  for i:= 1 to 10 do begin
    Write('Skriv heltal nr ', i, ': '); ReadLn(v1^[i]);
  end;
  WriteLn('Andra vektorn');
  for i:= 1 to 10 do begin
    Write('Skriv heltal nr ', i, ': '); ReadLn(v2^[i]);
  end;
  sum:= 0;
  for i:= 1 to 10 do
    sum:= sum + v1^[i] + v2^[i];
  WriteLn('Summan är ', sum);
  Dispose(v1);
  Dispose(v2);
end.

```

```

program Ovn_15_6;
(* Det är oklart vilket slags program läroboksförfattarna tänkt sig.
   Denna lösning hanterar en enkellänkad lista av heltal *)

uses CRT;

type
  Talpekare = ^Talnod;
  Talnod = record
    tal : Integer;
    next: Talpekare;
  end;

procedure Meny(var alt: Integer);
begin
  ClrScr;
  WriteLn('MENY':30);
  WriteLn;
  WriteLn('':10, '1. Skapa enkellänkad lista');
  WriteLn('':10, '2. Skriva ut alla tal i listan');
  WriteLn('':10, '0. Avsluta');
  WriteLn;
  repeat
    Write('':10, 'Välj alternativ: '); ReadLn(alt);
  until alt in [0..2];
end;

procedure RensaLista(var l: Talpekare);
var
  temp: Talpekare;
begin
  while l <> nil do begin
    temp:= l;
    l:= l^.next;
    Dispose(temp);
  end;
end;

procedure SkapaLista(var lista: Talpekare);
var
  i, antal: Integer;
  p: Talpekare;
begin
  RensaLista(lista);
  Randomize;
  Write('Ange hur många tal som ska genereras: ');
  ReadLn(antal);
  for i:= 1 to antal do begin
    New(p);
    p^.tal:= Random(1000); (* 0..999 *)
    p^.next:= lista;
    lista:= p;
  end;
end;

```

```

procedure SkrivLista(lista: Talpekare);
begin
  while (lista <> nil) do begin
    Write(lista^.tal:5);
    lista:= lista^.next
  end;
  WriteLn;
end;

var
  lista: Talpekare;
  alt : Integer;
begin
  lista:= nil;
  repeat
    Meny(alt);
    ClrScr;
    case alt of
      0: (* *) ;
      1: SkapaLista(lista);
      2: SkrivLista(lista);
    end;
    if alt <> 0 then begin
      Write('<Enter> för forts'); ReadLn;
    end;
  until alt = 0;
end.

```

Övning 15.7

Förutom proceduren Vand (se övningen), infoga följande i slutet av huvudprogrammet (i Pek1):

```

  WriteLn('Nu vänder vi listan');
  Vand(lista);
  PrintList(lista); (* Skriver ut igen *)
  Writeln('Tryck på Enter...'); (* Som förut *)

program Ovn_15_12; (* inbegriper även övning 11 *)
(* Använder här enbart Ltyp som datatyp för listan - den i boken
  använda typen pTyp behövs ej.
  Listhanteringen har mycket gemensamt med lösningen till övning
  15.6 ovan, med den skillnaden att vi här använder listhuvud.
  *)

type
  Ltyp = ^Dtyp;
  Dtyp = record
    tal : Integer;
    next: Ltyp;
  end;

procedure SkapaLista(var lista: Ltyp); (* Skapar listhuvud *)
begin
  new(lista);
  lista^.next:=nil;
end;

```

```

procedure FyllLista(lista: Ltyp);
var
  i, antal: Integer;
  p: Ltyp;
begin
  Randomize;
  Write('Ange hur många tal som ska lagras i listan: ');
  ReadLn(antal);
  for i:= 1 to antal do begin
    New(p);
    p^.tal:= Random(80); (* 0..79 *)
    p^.next:= lista^.next;
    lista^.next:= p;
  end;
end;

procedure SkrivLista(lista: Ltyp);
begin
  lista:= lista^.next; (* hoppa över listhuvudet *)
  while (lista <> nil) do begin
    Write(lista^.tal:5);
    lista:= lista^.next
  end;
  WriteLn;
end;

var
  lista: Ltyp;
begin
  SkapaLista(lista);
  FyllLista(lista);
  SkrivLista(lista);
end.

```

Övning 15.13

```

function Antal(lista: Ltyp): Word;
(* Här förutsätts att listan har listhuvud enligt läroboken.
  Läroboken använder en speciell typ (pTyp) och en lokal
  pekari variabel för att att löpa igenom listor, vilket dock inte
  behövs -- den extra typen är onödig och lokal pekari variabel
  behövs enbart om listan är var-deklarerad *)
var
  n: Word;
begin
  n := 0;
  lista := lista^.next;
  while lista <> nil do begin
    n:= n + 1;
    lista := lista^.next;
  end;
  Antal:= n;
end;

```

Övning 15.15

Det är enklare att lösa uppgiften med pekare i st f platsnummer (se övning 15.17 nedan):

```
function Mindre(n, m: Ltyp): Boolean;
begin
  Mindre := n^.tal < m^.tal;
end;
```

Övning 15.17

Vi löser uppgiften genom att komplettera övning 15.13 med procedurerna `BytTal` och `SorteraLista`. Dessutom har vi kompletterat huvudprogrammet.

I boken (övning 15.16 och 15.17) föreslås att man ska använda platsnumret när man ska byta 2 element med varandra. Detta är ineffektivt och klumpigt, eftersom det medför att man tvingas löpa igenom listan varje gång man vill åt ett element (funktionen `DatPek` i övning 5.14 har just till uppgift att leverera pekare till ett element med givet platsnummer).

Det är bättre, effektivare (och enklare) att strunta i platsnumren och hålla sig till pekare (man måste ju ändå använda pekarna vid utbytet). Följande `BytTal`-procedur använder därför pekare i stället för platsnummer (vi klarar oss då också med två i st för tre parametrar till `BytTal`).

Vi använder heller inte någon `Mindre`-funktion utan gör jämförelsen på vanligt sätt (att använda en sådan funktion med pekare som parametrar i st för bokens modell är i det närmaste trivialt, se lösning till 15.14 ovan och kommentaren i programlistningen nedan).

```
procedure BytTal(n, m: Ltyp);
var
  temp: Integer;
begin
  temp := n^.tal; n^.tal := m^.tal; m^.tal := temp;
end;

procedure SorteraLista(lista: Ltyp);
  (* Urvalssorterar genom att byta ut DATA-delen *)
var
  minP, p: Ltyp;
begin
  lista := lista^.next; (* hoppa över listhuvudet *)
  if lista = nil then Exit; (* Tom lista *)
  while lista^.next <> nil do begin
    minP := lista;
    p := lista^.next;
    while p <> nil do begin (* Sök reda på minsta element *)
      if p^.tal < minP^.tal then minP := p;
      (* Med Mindre-funktion gör vi så här i stället: *)
      (* if Mindre(p, minP) then minP := p; *)
      p := p^.next;
    end (* while *);
    if minP <> lista then (* Byt innehåll *)
      BytTal(lista, minP);
    lista := lista^.next;
  end;
end;
```

```
var
  lista: Ltyp;
begin
  SkapaLista(lista);
  FyllLista(lista);
  WriteLn('Osorterad lista:');
  SkrivLista(lista);
  SorteraLista(lista);
  WriteLn('Sorterad lista:');
  SkrivLista(lista);
end.
```