

8 Übung zu Informatik zum 17.12.2009 Blatt 8

8.1

```
program abundant;
var i:Integer; kontr:Integer;
function Teilersumme(zahl:Integer):Integer; {Teilersumme einer Zahl ermitteln}
var d:Integer; erg:Integer=0;
begin
  for d:=1 to zahl div 2 do
    begin
      if zahl mod d = 0 then erg := erg + d; {Gehe alle Zahlen durch und summiere
        Zahlen, die Eingabezahl restlos teilen}
      end;
    Teilersumme:=erg {Gebe Teilersummer zurück}
  end;
begin
  for i:=2 to 1000 do
    begin
      kontr := Teilersumme(i); {Ermittel Teilersumme für aktuelles i}
      if kontr=i then {Teilersumme einer Zahl = Zahl -> vollkommen}
        writeln(i, '_ist_vollkommene_Zahl')
      else
        begin
          if kontr>i then {Teilersumme einer Zahl > Zahl -> abundant}
            writeln(i, '_ist_abundante_Zahl')
          else {Teilersumme einer Zahl < Zahl -> defizient}
            writeln(i, '_ist_defiziente_Zahl')
          end;
        end;
      end;
    readln
  end.
end.
```

8.2

```
program wallisch;  
  var n:Integer;  
  function wallisch:Real; { Gibt pi/halbe nach Wallischem Produkt aus}  
  var i:Integer; ausgabe:Real;  
  begin  
    if n > 0 then ausgabe:=2; { 0 Durchläufe ergibt "0", 1 Durchlauf "2"}  
    for i:=2 to n do  
      begin  
        if i mod 2 = 0 then  
          ausgabe := ausgabe * (i/(i+1)) { Gerader Durchlauf}  
        else  
          ausgabe := ausgabe * ((i+1)/i); { Ungerader Durchlauf}  
        end;  
        wallisch:=ausgabe  
      end;  
    end;  
  begin  
    writeln('Geben_Sie_bitte_eine_Zahl_ein , die die Anzahl der Faktoren darstellt ,  
           die im wallischen Produkt berechnet werden sollen . '); { Begrüßung/Anweisung}  
    write('n=');  
    readln(n); { Eingabe der Faktor-Anzahl}  
    writeln('pi/2_ist_ungefaehr_', wallisch:1:7); { Ausgabe und Warten auf Return}  
    readln  
  end.  
end.
```

```

program siebensegment;
  var zifferfolge:String;
procedure zeile(c:Integer); {Schreibt die Zeile c der Siebensegmentzahlen in die
    Kommando-Zeile}
  var i:Integer; teil:string;
begin
  for i:=1 to length(zifferfolge) do {Gehe jede Zahl durch}
  begin
    teil:=copy(Zifferfolge,i,1); {Nur das i-te Zeichen der eingegebenen Zahl}
    if (teil='0') or (teil='1') or (teil='2') or (teil='3') or (teil='4') or (teil='5')
      or (teil='6') or (teil='7') or (teil='8') or (teil='9') then {Wenn eine
      Ziffer, dann}
    begin
      case c of {Optimierte Übersetzung einer Ziffer zu ihren jeweiligen 3
        Siebensegment-Zeichen, wie definiert. Zeilenunterscheidung}
        1: if (teil='1')or (teil='4') then write(' _ _ _') else write(' — ');
        2: if (teil='5')or (teil='6') then write(' | _ _') else if (teil='4') or (
          teil='8') or (teil='9')or (teil='0') then write(' | _ |') else write(' _ _ |
          ');
        3: if (teil='0') or (teil='1') or (teil='7') then write(' _ _ _') else write('
          — ');
        4: if (teil='2') then write(' | _ _') else if (teil='6') or (teil='8') or (
          teil='0') then write(' | _ |') else write(' _ _ |');
        5: if (teil='1') or (teil='4') or (teil='7') then write(' _ _ _') else write
          (' — ');
      end;
      write(' _') {Leerzeichen zwischen Siebensegmentzahlen}
    end;
  end;
  writeln('') {Nächste Zeile der Siebensegmentzahl}
end;
procedure eingabe;
begin
  writeln('Geben Sie eine Ziffernfolge mit max. 64 Zeichen fuer die
    Siebensegmentdarstellung ein oder "quit" um zu beenden'); {Begrüßung}
  readln(zifferfolge);
  if length(zifferfolge) <65 then
  begin
    if not (zifferfolge = 'quit') then {quit beendet Eingabe-Schleife}
    begin
      zeile(1); {Jede der 5 Zeilen der Siebensegmentzahl ausgeben}
      zeile(2);
      zeile(3);
      zeile(4);
      zeile(5);
      eingabe
    end;
  end
  else
  begin
    writeln('Ziffernfolge zu lang!'); {Wenn >64 Zeichen eingegeben wurden,
      Fehlerausgabe}
    eingabe
  end;
end;
begin
  eingabe
end.

```