1 Wprowadzenie

```
program test;

begin
  writeln('Witaj');
end.
```

Polecenie **program**.
Różnice między poleceniami **write** i **writeln**.

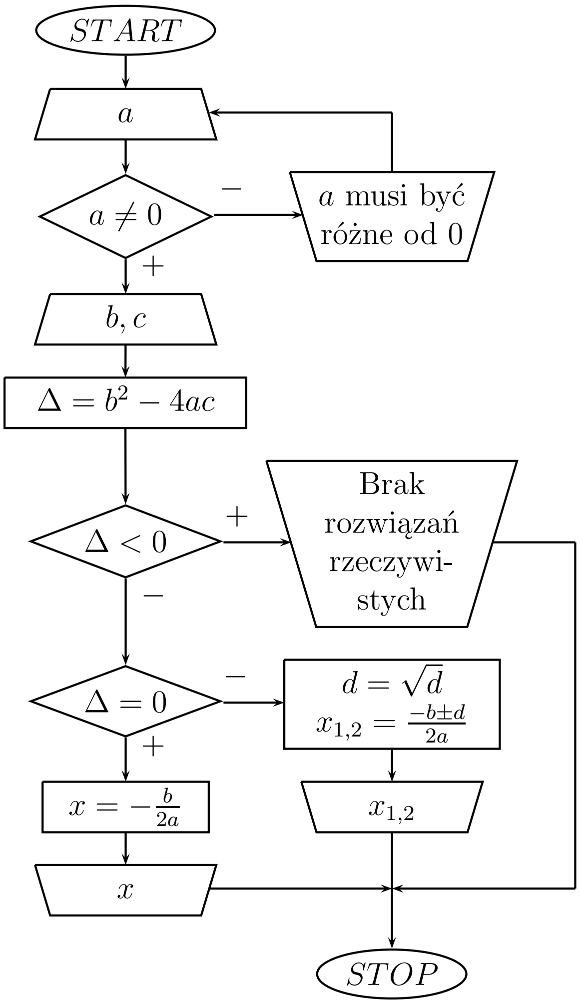
```
program test2;

var a:string;

begin
  write('Podaj imie: ');
  readln(a);
  writeln('Witaj ',a);
end.
```

Porównanie **read** i **readln**.
Zwrócenie uwagi na spacje występujące w poleceniach **write** i **writeln**.

```
program test3;

var a,b,c:integer;
    d:real;

begin
  writeln('Podaj dwie liczby calkowite');
  write('a=');
  readln(a);
  write('b=');
  readln(b);
  c:=a+b;
  writeln('Suma wynosi ',c);
  c:=a-b;
  writeln('Roznica wynosi ',c);
  c:=a*b;
  writeln('Iloczyn wynosi ',c);
  d:=a/b;
  writeln('Iloraz wynosi ',d);
end.
```

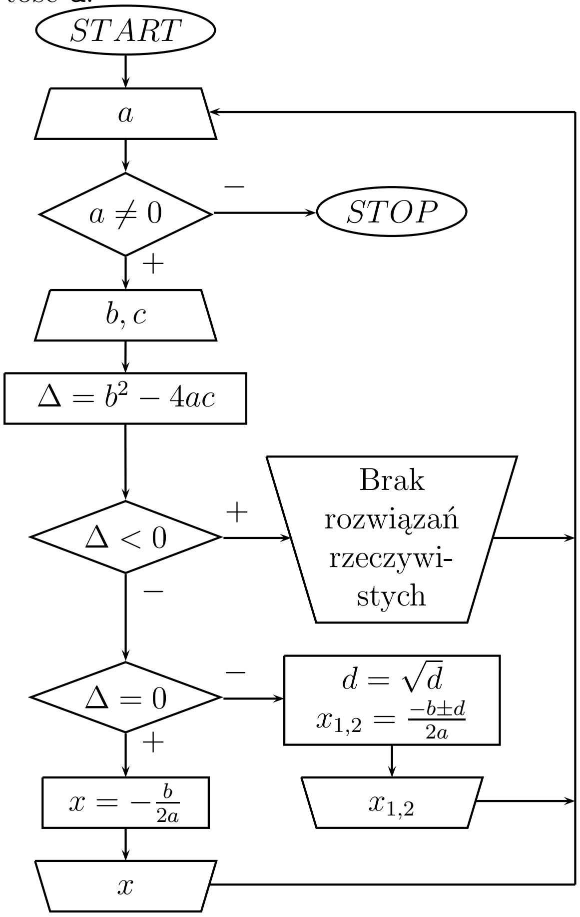
2 Rozwiązywanie trójmianu kwadratowego

Pętla sterująca (`repeat–until`), aby zmienna `a` była różna od zera. Dopiero, gdy zostanie tak zdefiniowana program liczy dla jednego trójmianu kwadratowego.




```
program trojmian;

var a,b,c:integer;
    d,x:real;

begin
  writeln('Podaj calkowite
     wsp. trojmianu
     kwadratowego');
  repeat
    write('a=');
    readln(a);
    if (a=0) then
       writeln('wsp. a musi
       byc rozny od zera!');
  until (a<>0);
  write('b=');
  readln(b);
  write('c=');
  readln(c);
  d:=b*b-4*a*c;
  if (d<0) then
    writeln('Brak pierwiastkow
        rzeczywistych')
  else
    if (d=0) then
      begin
        x:=-b/(2*a);
        writeln('Rozw.:  ',x);
      end
    else
      begin
        d:=sqrt(d);
        x:=(-b-d)/(2*a);
        writeln('Rozw.:  ',x);
        x:=(-b+d)/(2*a);
        writeln('Rozw.:  ',x);
      end;
  readln;
end.
```

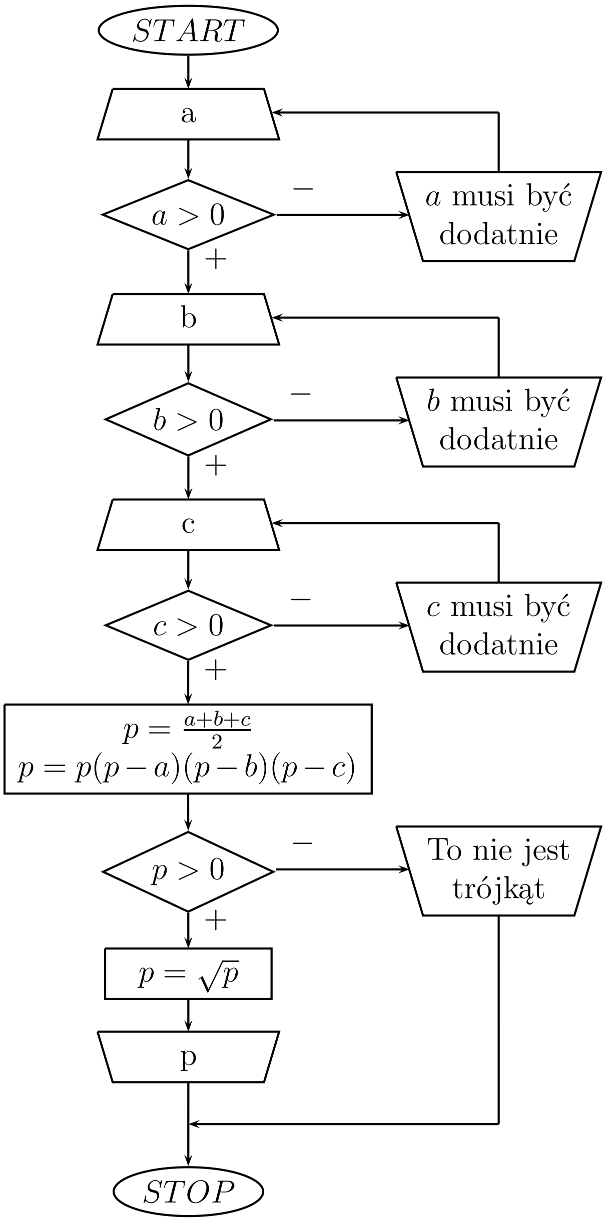
Pętla sterująca (`while`), która kończy działanie programu, gdy zmienna `a` jest równa zero. W pozostałych przypadkach definiowane są pozostałe dane, rozwiązywany trójmian kwadratowy, a następnie jest pytanie o kolejną wartość `a`.




```
program trojmian;

var a,b,c:integer;
    d,x:real;

begin
  writeln('Podaj calkowite
     wsp. trojmianu
     kwadratowego');
  write('Podaj wsp. a (=0 -
     koniec) a=');
  readln(a);
  while(a<>0) do
    begin
      write('b=');
      readln(b);
      write('c=');
      readln(c);
      d:=b*b-4*a*c;
      if (d<0) then
        writeln('Brak
           pierwiastkow
           rzeczywistych')
      else
        if (d=0) then
          begin
            x:=-b/(2*a);
            writeln('Rozw.:
               ',x);
          end
        else
          begin
            d:=sqrt(d);
            x:=(-b-d)/(2*a);
            writeln('Rozw.:
               ',x);
            x:=(-b+d)/(2*a);
            writeln('Rozw.:
               ',x);
          end;
      write('Podaj wsp. a (=0
         - koniec) a=');
      readln(a);
    end;
  readln;
end.
```

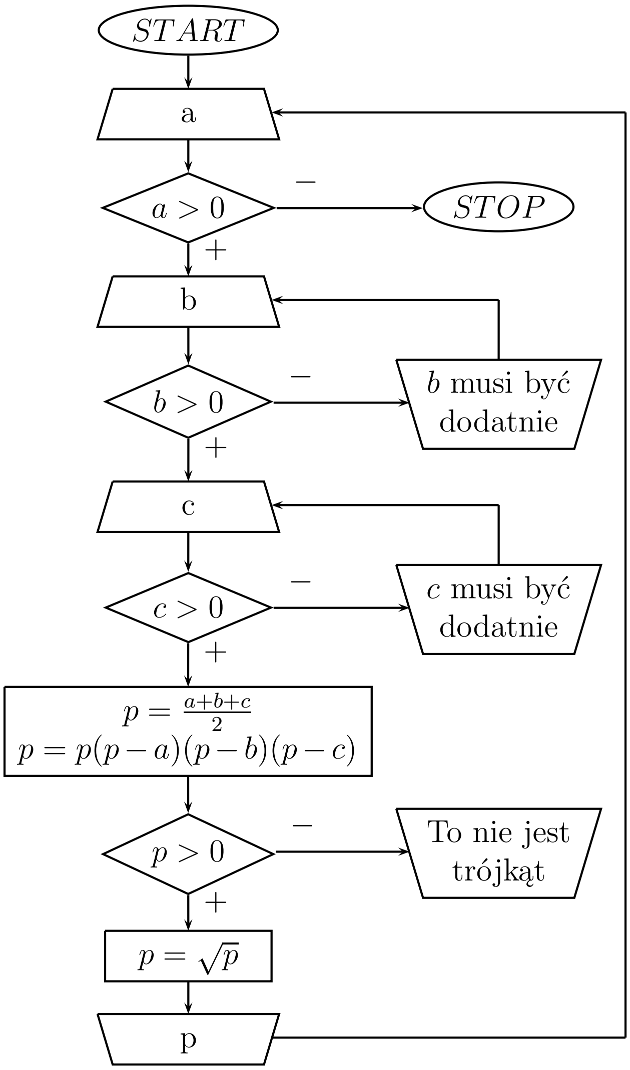
3 Wyznaczanie pola trójkąta - wzór Herona

Wersja programu, która wymusza podanie dodatnich wartości długości boków dla jednego trójkąta. Po obliczeniach program się kończy.



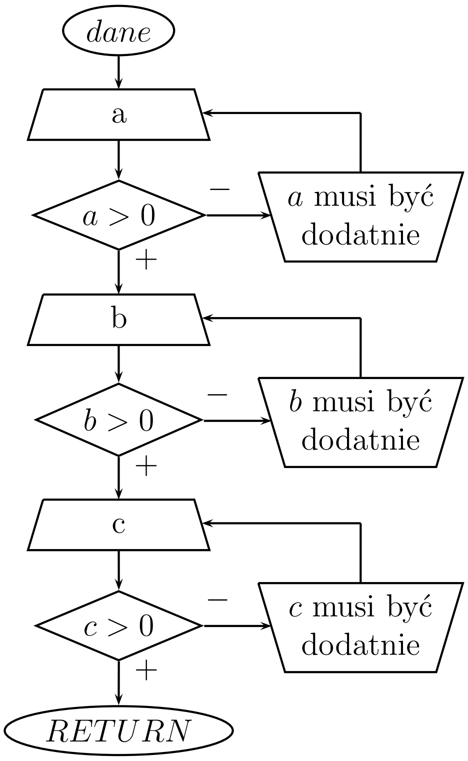

```
program heron;

var a,b,c:integer;
    p:real;

begin
  writeln('Podaj dlugosci
     bokow trojkata');
  repeat
    write('a=');
    readln(a);
    if (a=<0) then
       writeln('dlugosc boku
       musi byc dodatnia');
  until (a>0);
  repeat
    write('b=');
    readln(b);
    if (b=<0) then
       writeln('dlugosc boku
       musi byc dodatnia');
  until (b>0);
  repeat
    write('c=');
    readln(c);
    if (c=<0) then
       writeln('dlugosc boku
       musi byc dodatnia');
  until (c>0);
  p:=(a+b+c)/2;
  p:=p*(p-a)*(p-b)*(p-c);
  if (p>0) then
    begin
      p:=sqrt(p);
      writeln('Pole trojkata
         wynosi: ',p);
    end
  else
    writeln('Takie odcinki nie
       tworza trojkata');
  readln;
end.
```

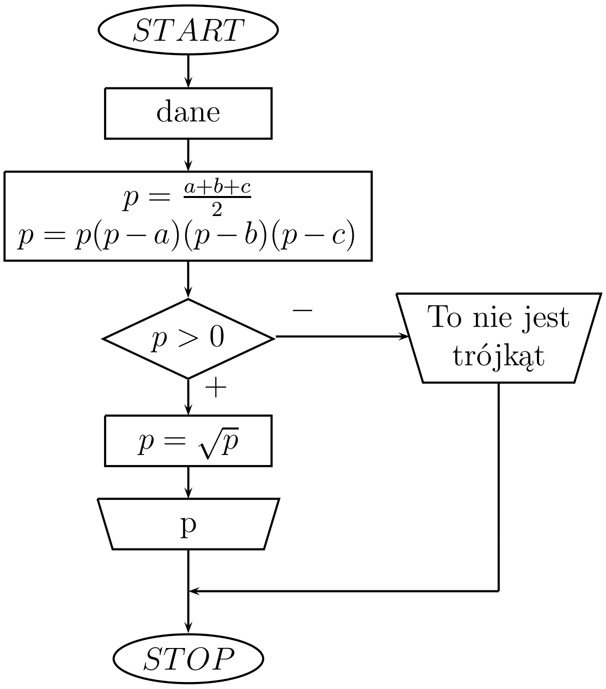
Wersja programu,z pętlą `while`, w której obliczane jest pole dla trójkątów tak długo, aż poda się niedodatnią długość pierwszego boku (długości pozostałych boków muszą być dodatnie) .




```
program heron;

var a,b,c:integer;
    p:real;

begin
 writeln('Podaj dlugosci bokow
     trojkata');
 write('Bok a (<=0 - koniec)
    a=');
 readln(a);
 while(a>0) do
  begin
   repeat
    write('b=');
    readln(b);
    if (b=<0) then
       writeln('dlugosc boku
       musi byc dodatnia');
   until (b>0);
   repeat
    write('c=');
    readln(c);
    if (c=<0) then
       writeln('dlugosc boku
       musi byc dodatnia');
   until (c>0);
   p:=(a+b+c)/2;
   p:=p*(p-a)*(p-b)*(p-c);
   if (p>0) then
    begin
     p:=sqrt(p);
     writeln('Pole trojkata
        wynosi: ',p);
    end
   else
    writeln('Takie odcinki nie
        tworza trojkata');
   write('Bok a (<=0 -
      koniec) a=');
   readln(a);
  end;
 readln;
end.
```

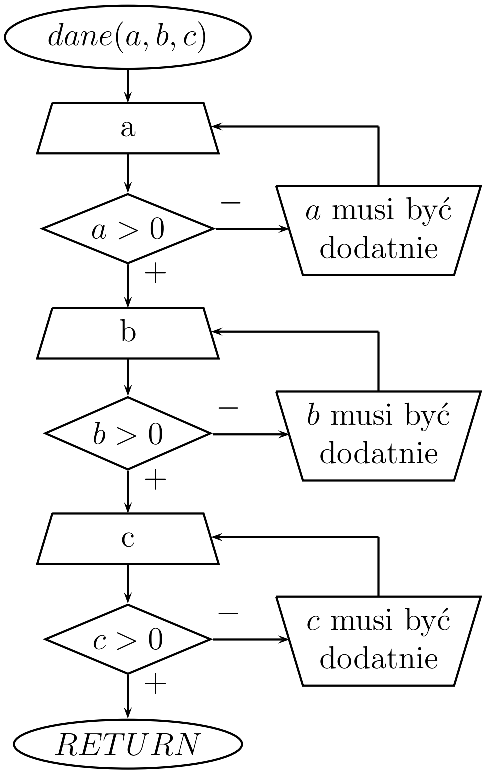
Modyfikacja pierwszego programu polegająca na zdefiniowaniu procedury działającej na zmiennych globalnych. Definicja zmiennych `a`, `b`, `c` w programie głównym musi być przed definicją procedury.

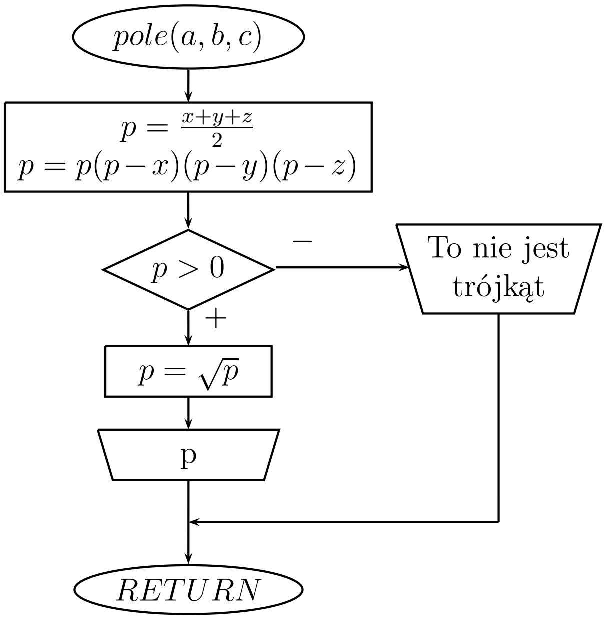


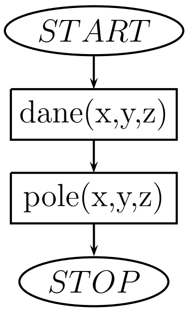



```
program heron;

var a,b,c:integer;
    p:real;

procedure dane;
begin
  writeln('Podaj dlugosci
     bokow trojkata');
  repeat
    write('a=');
    readln(a);
    if (a=<0) then
       writeln('dlugosc boku
       musi byc dodatnia');
  until (a>0);
  repeat
    write('b=');
    readln(b);
    if (b=<0) then
       writeln('dlugosc boku
       musi byc dodatnia');
  until (b>0);
  repeat
    write('c=');
    readln(c);
  if (c=<0) then
     writeln('dlugosc boku
     musi byc dodatnia');
  until (c>0);
end;

begin
  dane;
  p:=(a+b+c)/2;
  p:=p*(p-a)*(p-b)*(p-c);
  if (p>0) then
    begin
      p:=sqrt(p);
      writeln('Pole trojkata
         wynosi: ',p);
    end
  else
    writeln('Takie odcinki nie
        tworza trojkata');
  readln;
end.
```

Wersja programu ze zdefiniowanymi procedurami działającymi na zmiennych lokalnych (nazewnictwo zmiennych w procedurach i programie głównych jest niezależne; muszą być tylko takich samych typów). Stąd definicja zmiennych w programie może wystąpić po definicjach procedur.

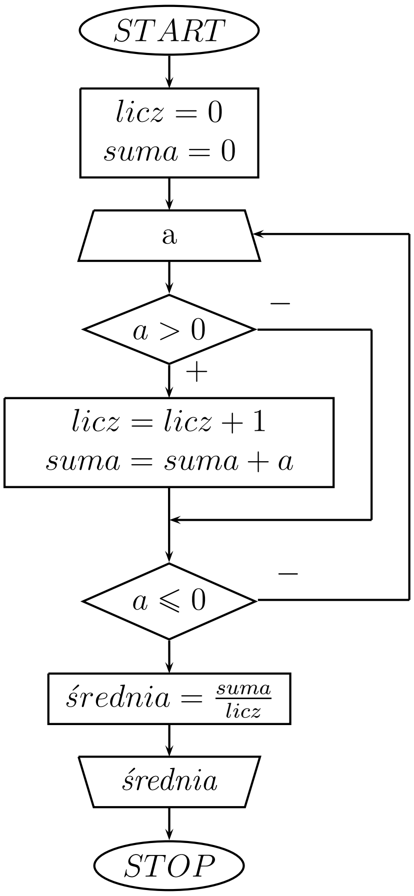






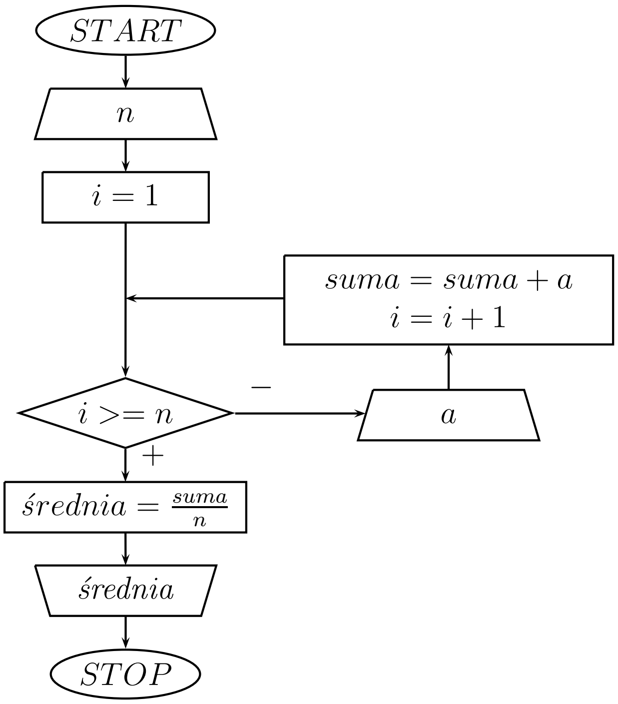

```
program heron;
procedure dane(var
    a,b,c:integer);
begin
  writeln('Podaj dlugosci
     bokow trojkata');
  repeat
    write('a=');
    readln(a);
    if (a=<0) then
       writeln('dlugosc boku
       musi byc dodatnia');
  until (a>0);
  repeat
    write('b=');
    readln(b);
    if (b=<0) then
       writeln('dlugosc boku
       musi byc dodatnia');
  until (b>0);
  repeat
    write('c=');
    readln(c);
  if (c=<0) then
     writeln('dlugosc boku
     musi byc dodatnia');
  until (c>0);
end;
procedure pole(a,b,c:integer);
var p:real;
begin
  p:=(a+b+c)/2;
  p:=p*(p-a)*(p-b)*(p-c);
  if (p>0) then
    begin
      p:=sqrt(p);
      writeln('Pole trojkata
         wynosi: ',p);
    end
  else
    writeln('Takie odcinki nie
        tworza trojkata');
end;
var x,y,z:integer;
begin
  dane(x,y,z);
  pole(x,y,z);
  readln;
end.
```

4 Wyznaczanie średniej z liczb



Program wyznacza średnią z dodatnich liczb podawanych w trakcie uruchomienia programu. Podanie liczby równej zero lub ujemnej jest informacją o zakończeniu podawania danych

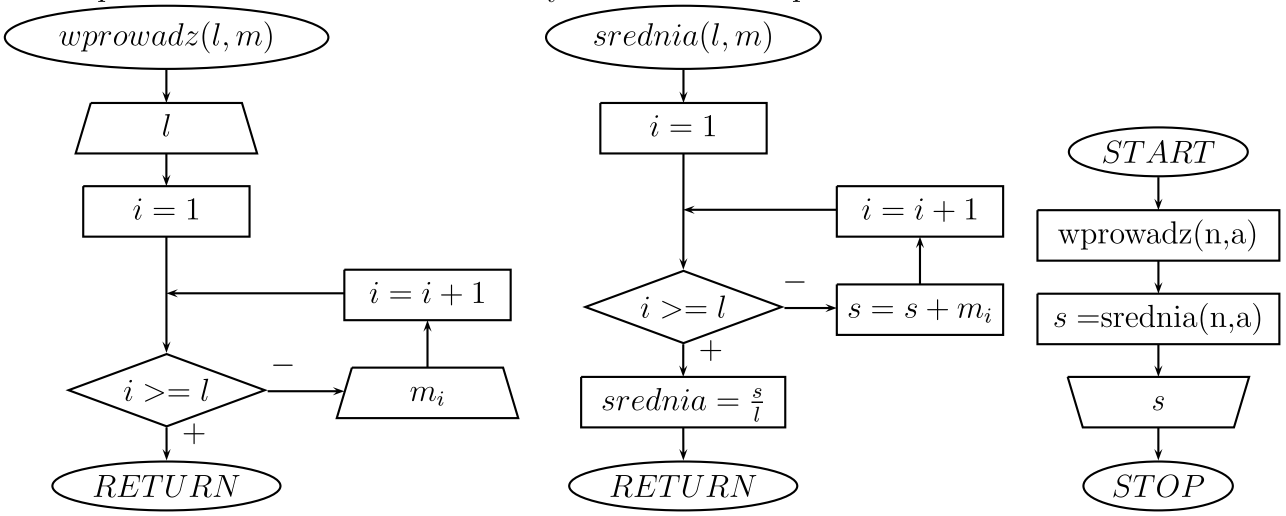
```
program srednia;

var a,s:real;
    licz:integer;

begin
  writeln('Program liczy srednia z liczb dodatnich (a<=0 -
     koniec)');
  licz:=0;
  suma:=0.0;
  repeat
    writeln('Podaj liczbe: ');
    readln(a);
    if (a>0) then
      begin
        suma:=suma+a;
        licz:=licz+1;
      end;
  until (a<=0);
  s:=s/n;
  writeln('Srednia liczb wynosi: ',s);
  readln;
end.
```



Modyfikacja programu, w której użytkownik na początku definiuje ilość podawanych liczb, a dopiero później je podaje.

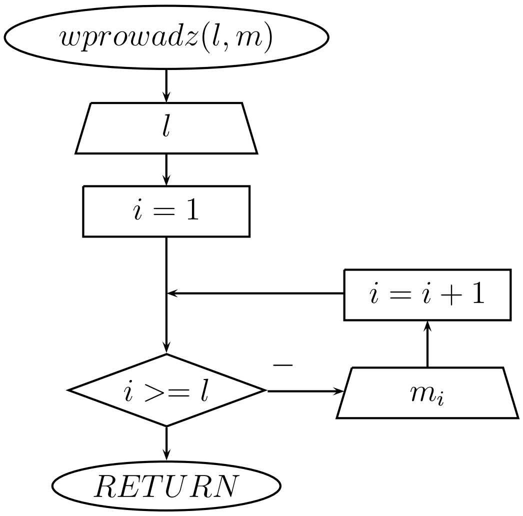
```
program srednia2;

var a,s:real;
    n,i:integer;

begin
  writeln('Program liczy srednia z liczb');
  writeln('Podaj ilosc liczb: ');
  readln(n);
  s:=0.0;
  for i:=1 to n do
    begin
      write('Podaj liczbe nr ',i,': ');
      readln(a);
      s:=s+a;
    end;
  s:=s/n;
  writeln('Srednia liczb wynosi: ',s);
  readln;
end.
```

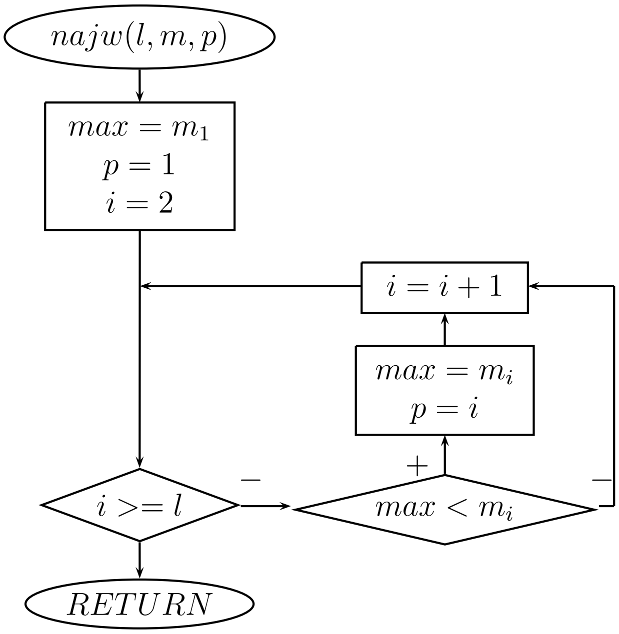
Program ze zmiennymi tablicowymi. Umożliwia to oddzielenie procesu wprowadzania danych od ich przetwarzania i w efekcie końcowym zdefiniowanie procedur.

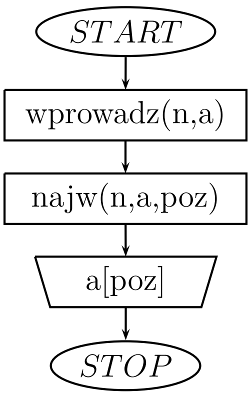



```
program srednia_m;

type lista=array[1..100] of real;
var a:lista;
    s:real;
    n:integer;

procedure wprowadz(var l:integer;var m:lista);
var i:integer;
begin
  write('Podaj ilosc liczb: ');
  readln(l);
  for i:=1 to l do
    begin
      write('Podaj liczbe nr ',i,': ');
      readln(m[i]);
    end;
end;
function srednia(l:integer;m:lista):real;
var i:integer;
    s:real;
begin
  for i:=1 to l do
    s:=s+m[i];
  srednia:=s/l;
end;

begin
  writeln('Program liczy srednia podanych liczb');
  wprowadz(n,a);
  s:=srednia(n,a);
  writeln('Srednia liczb wynosi: ',s);
  readln;
end.
```

5 Wyszukiwanie największej wartości

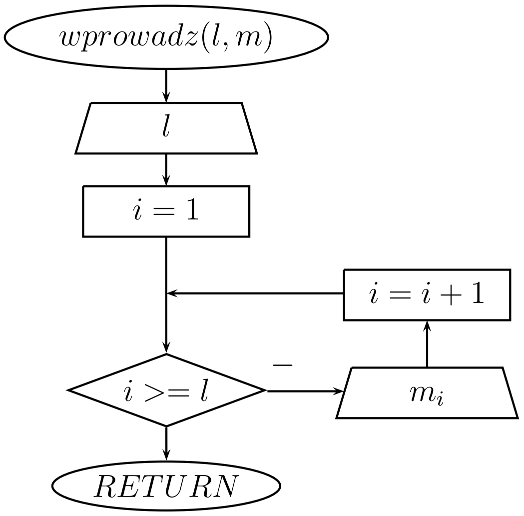


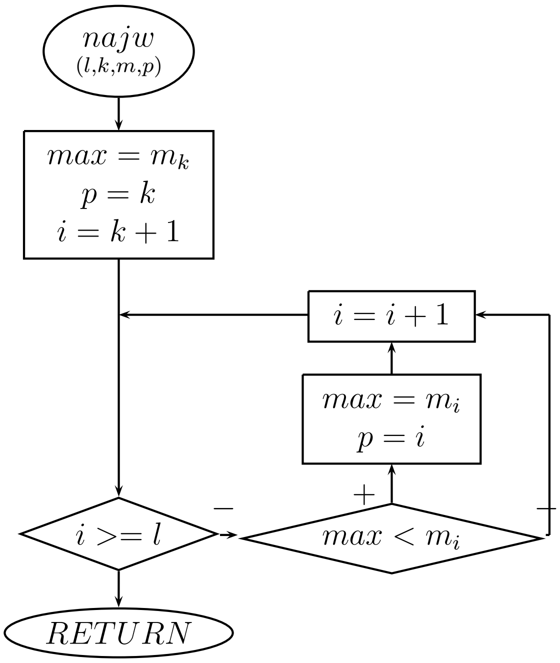


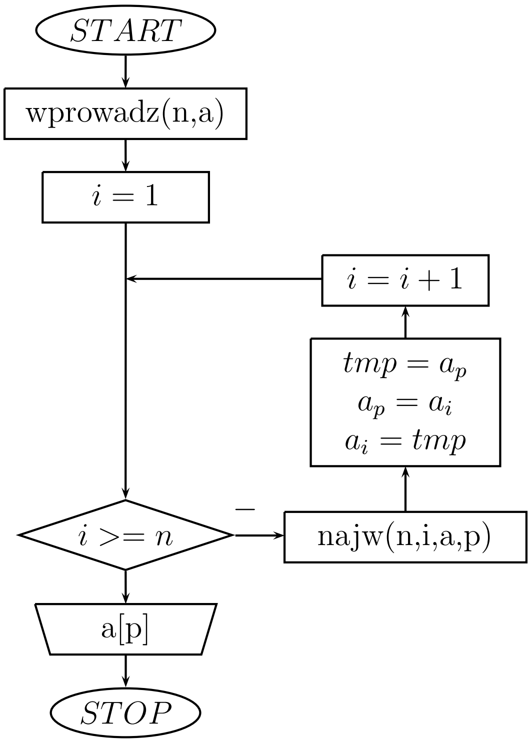



```
program najwieksza;

type lista=array[1..100] of
   real;

procedure wprowadz(var
   l:integer;var m:lista);
var i:integer;
begin
  write('Podaj ilosc liczb:
      ');
  readln(l);
  for i:=1 to l do
    begin
      write('Podaj liczbe nr
          ',i,': ');
      readln(m[i]);
    end;
end;
procedure
   najw(l:integer;m:lista;var
   p:integer);
var i:integer;
     max:real;
begin
  max:=m[1];
  p:=1;
  for i:=2 to l do
    if (max<m[i]) then
      begin
        max:=m[i];
        p:=i;
      end;
end;
var n,poz:integer;
     a:tablica;
begin
  writeln('Program wyszukuje
      najwieksza liczbe sposrod
      podanych');
  wprowadz(n,a);
  najw(n,a,poz);
  writeln('Najwieksza liczba
      to ',a[poz],' na pozycji
      ',poz);
  readln.
end.
```

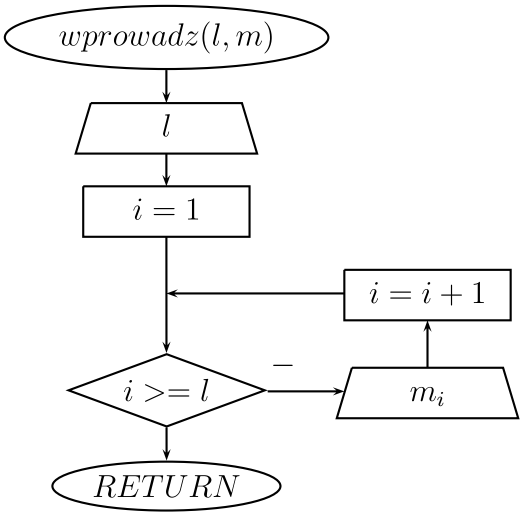
6 Sortowanie z wyborem największej wartości

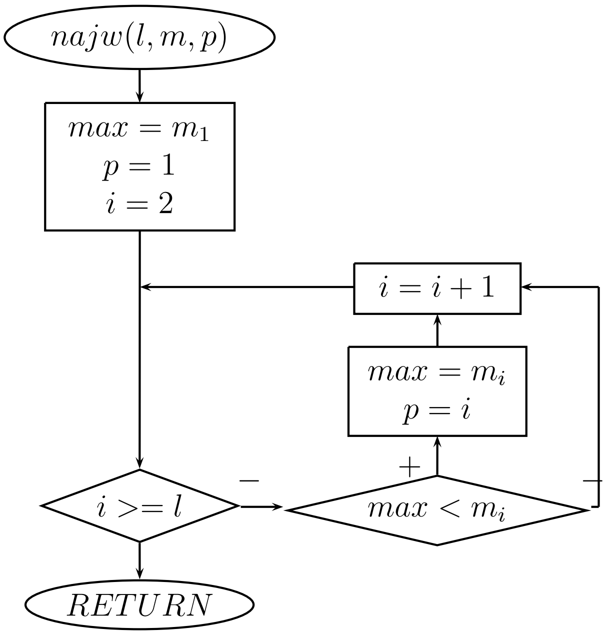


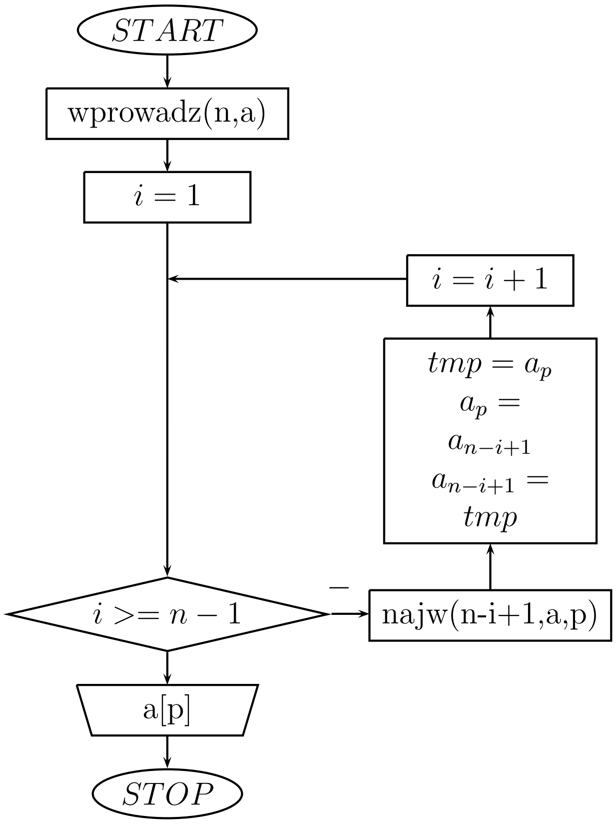





```
program sortuj;
type lista=array[1..100] of
   real;
procedure wprowadz(var
   l:integer;var m:lista);
var i:integer;
begin
  write('Podaj ilosc liczb:
     ');
  readln(l);
  for i:=1 to l do
    begin
      write('Podaj liczbe nr
         ',i,': ');
      readln(m[i]);
    end;
end;
procedure najw(l,k:integer;
   m:lista;var p:integer);
var i:integer;
    max:real;
begin
  max:=m[k];
  p:=k;
  for i:=k+1 to l do
    if (max<m[i]) then
      begin
        max:=m[i];
        p:=i;
      end;
end;
var n,j,p:integer;
    tmp:real;
    a:tablica;
begin
  writeln('Program wyszukuje
     najwieksza liczbe sposrod
      podanych');
  wprowadz(n,a);
  for j:=1 to n-1 do
    begin
      najw(n,j,a,p);
      tmp:=a[p];
      a[p]:=a[j];
      a[j]:=tmp;
    end;
  for j:=1 to n do
    write(a[j],'    ');
  readln.
end.
```








```
program sortuj;

type lista=array[1..100] of
    real;

procedure wprowadz(var
    l:integer;var m:lista);
var i:integer;
begin
  write('Podaj ilosc liczb:
      ');
  readln(l);
  for i:=1 to l do
    begin
      write('Podaj liczbe nr
          ',i,': ');
      readln(m[i]);
    end;
end;
procedure najw(l:integer;
    m:lista;var p:integer);
var i:integer;
    max:real;
begin
  max:=m[1];
  p:=1;
  for i:=2 to l do
    if (max<m[i]) then
      begin
        max:=m[i];
        p:=i;
      end;
end;
var n,i,p:integer;
    tmp:real;
    a:tablica;
begin
  writeln('Program wyszukuje
      najwieksza liczbe sposrod
      podanych');
  wprowadz(n,a);
  for i:=1 to n-1 do
    begin
      najw(n-i+1,a,p);
      tmp:=a[p];
      a[p]:=a[n-i+1];
      a[n-i+1]:=tmp;
    end;
  for j:=1 to n do
    write(a[j],'     ');
  readln;
end.
```

7 Sortowanie metodą bąbelkową

```
program bab;

type lista=array[1..100] of
    real;

procedure wprowadz(var
    l:integer;var m:lista);
var i:integer;
begin
  repeat
    write('Podaj ilosc liczb:
        ');
    readln(l);
  until (l>0) and (l<=100);
  for i:=1 to l do
    begin
      write('Podaj liczbe nr
          ',i,': ');
      readln(m[i]);
    end;
end;

procedure
    sort_bab(l:integer;var
    m:lista);
var i,j:integer;
begin
  for i:=1 to l-1 do
    for j:=1 to l-1 do
      if m[j]<m[j+1] then
        begin
          tmp:=m[j];
          m[j]:=m[j+1];
          m[j+1]:=tmp;
        end;
end;

procedure
    wypisz(l:integer,m:lista);
var i:integer;
begin
  for i:=1 to l do
    write(m[i],'     ');
  readln;
end;

begin
  wprowadz(n,a);
  sort_bab(n,a);
  wypisz(n,a);
end.
```

8 Dodawanie i mnożenie macierzy

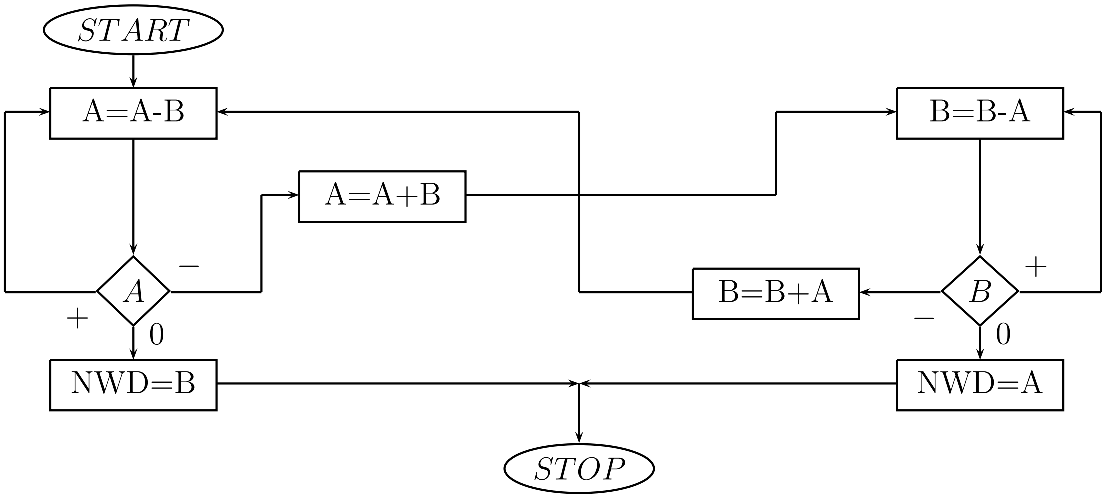
```
program macierze;
const max=10;
type macierz=array[1..max,1..max] of integer;
procedure wpisz(var w,k:integer;max:integer;var m:macierz};
  var i,j:integer;
  begin
    repeat
      write('Podaj ilosc wierszy ');
      readln(w);
    until (w>0) and (w<=max);
    repeat
      write('Podaj ilosc kolumn ');
      readln(k);
    until (k>0) and (k<=max);
    for i:=1 to w do
      for j:=1 to k do
        begin
          write('Element (',i,',',j,')=');
          readln(m[i,j]);
        end;
  end;
procedure wypisz(w,k:integer;m:macierz);
  var i,j:integer;
  begin
    for i:=1 to w do
      begin
        for j:=1 to k do
          write(m[i,j],'   ');
        writeln;
      end;
  end;
procedure dodaj(w,k:integer;m1,m2:macierz;var m3:macierz);
  var i,j:integer;
  begin
    for i:=1 to w do
      for j:=1 to k do
        m3[i,j]:=m1[i,j]+m2[i,j];
  end;
procedure mnoz(w,k,l:integer;m1,m2:macierz;var m3:macierz);
  var i,j,s:integer;
  begin
    for i:=1 to w do
      for j:=1 to k do
        begin
          m3[i,j]:=0;
          for s:=1 to l do
            m3[i,j]:=m3[i,j]+m1[i,s]*m2[s,j];
        end;
  end;
var a,b,c:macierz;
    odp,lw1,lw2,lk1,lk2:integer;
```

```pascal
    test:boolean;
begin
  writeln('Program dziala na macierzach');
  repeat
    write(Co chcesz zrobic (1-dodawanie, 2-mnozenie)? ');
    readln(odp);
  until (odp=1) or (odp=2);
  wpisz(lw1,lk1,max,a);
  repeat
    wpisz(lw2,lk2,max,b);
    if odp=1 then
      test:=(lw1=lw2) and (lk1=lk2)
    else
      test:=lk1=lw2;
    if not(test) then
      write('Nie mozna przeprowadzic operacji ');
    if (odp=1) then
      writeln('dodawania')
    else
      writeln('mnozenia');
  until (test);
  if (odp=1) then
    dodaj(lw1,lk1,a,b,c)
  else
    mnoz(lw1,lk2,lw2,a,b,c);
  wypisz(lw1,lk1,c);
end.
```

9 Zamiana liczb arabskich na rzymskie

```
program arnarom;

const
     tys='M'; pset='D'; setk='C';
          pd='L'; dzie='X'; pie='V';
          jed='I';

var m,n:integer;

procedure romseg(s:integer;
    var r:integer);
 var z1,z2,z3:char;
 begin
  if s=100 then
   begin
    z1:=tys;
    z2:=setk;
    z3:=pset
   end
  else
   if s=10 then
    begin
     z1:=setk;
     z2:=dzie;
     z3:=pd
    end
   else
    if s=1 then
     begin
      z1:=dzie;
      z2:=jed;
      z3:=pie
     end
    else
     begin
      writeln('zla skala w
         ROMSEG !');
      halt
     end;
  while r>=10*s do
   begin
    write(z1);
    r:=r-10*s
   end;
  if r>=9*s then
   begin
    write(z2,z1);
    r:=r-9*s
   end
  else
   if r>=5*s then
    begin
     write(z3);
     r:=r-5*s
    end
   else
    if r>=4*s then
     begin
      write(z2,z3);
      r:=r-4*s
     end
 end {romseg};

begin
 repeat
  writeln;
  write('podaj liczbe
     naturalna, n=');
  readln(n);
  m:=n;
  if (n>0) and (n<maxint) then
   begin
    write('zapis rzymski : ');
    romseg(100,n);
    romseg( 10,n);
    romseg(  1,n);
    while n>=1 do
     begin
      write(jed);
      n:=n-1
     end;
    writeln
   end
 until m<=0
end.
```

10 Największy wspólny dzielnik




```
program ndwpr;

var a,b,ar,ndw,br:integer;
label 9999;

begin
  repeat
    writeln('podaj pare liczb
       calkowitych');
    readln(a,b);
  until (a>0) and (b>0);
  ar:=a;
  br:=b;

9999:
  repeat
    a:=a-b;
  until (a<=0);
  if (a<0) then
    begin
      a:=a+b;
      repeat
        b:=b-a;
      until(b<=0);
      if (b<0) then
        begin
          b:=b+a;
          goto 9999;
        end
      else
        ndw:=a;
    end
  else
    ndw:=b;
  writeln('NDW(',ar,',',br,
      ')=',ndw);
end.
```

Program ze zdefiniowaną procedurą rekurencyjną

```
program ndwprek;

var n,m,ndwnm,nr,mr:integer;

procedure ndwprek(var
    a,b,ndw:integer);
begin
  repeat
    a:=a-b;
  until (a<=0);
  if (a<0) then
    begin
      a:=a+b;
      ndwprek(b,a,ndw);
    end
  else
    ndw:=b;
end;

begin
  repeat
    writeln('podaj pare liczb
       calkowitych');
    readln(n,m);
  until (n>0) and (m>0);
  nr:=n;
  mr:=m;
  ndwprek(n,m,ndwnm);
  writeln('NDW(',nr,',',mr,
      ')=',ndwnm);
end.
```

Program ze zdefiniowaną funkcją rekurencyjną

```
program ndwfr;

var n,m,nr,mr:integer;

function ndwfrek(var
    a,b:integer):integer;
begin
  repeat
    a:=a-b;
  until (a<=0);
  if (a<0) then
    begin
      a:=a+b;
      ndwfrek:=ndwfrek(b,a);
    end
  else
    ndwfrek:=b;
end;

begin
  repeat
    writeln('podaj pare liczb
       calkowitych');
    readln(n,m);
  until (n>0) and (m>0);
  nr:=n;
  mr:=m;
  writeln('NDW(',nr,',',mr,
      ')=',ndwfrek(n,m));
end.
```

11 Płyty

Program, dający możliwość wpisania informacji o jednej płycie CD oraz wypisania na ekranie jej zawartości: tj. tytułu i wykonawcy albumu oraz wszystkich piosenek (łącznie z całkowitym czasem całej płyty).

```
program plyty;

type ut=record
      tyt:string;
      min,sek:integer;
      end;
type cd=record
      ptyt,wyk:string;
      n:integer;
      pios:array [1..10] of ut;
      end;

procedure wpisz(var p:cd);
var i:integer;
begin
 with p do
  begin
   write('Podaj tytul
      plyty:');
   readln(ptyt);
   write('Podaj wykonawce: ');
   readln(wyk);
   repeat
    write('Podaj ilosc utworow
       (max. 10): ');
    readln(n);
   until (n>0) and (n<=10);
   for i:=1 to n do
    with pios[i] do
     begin
      write(i:2,'. Tytul
         utworu: ');
      readln(tyt);
      write('    Czas trwania
         (min): ');
      readln(min);
      write('
         (sek): ');
      readln(sek);
     end;
  end;
end;

procedure wypisz(p:cd);
var i,tm,ts:integer;
begin
 with p do
  begin
   writeln('Wykonawca:
      ',wyk);
   writeln('Tytul plyty:
      ',ptyt);
   tm:=0;
   ts:=0;
   for i:=1 to n do
    with pios[i] do
     begin
      writeln(i:2,'.
         ',tyt:20,'
         ',min:2,':',sek:2);
      tm:=tm+min;
      ts:=ts+sek;
     end;
   tm:=tm+(ts div 60);
   ts:=ts-(ts div 60)*60;
   writeln('                TOTAL
       TIME: ',tm:2,':',ts:2);
  end;
end;

var krazek:cd;

begin
 wpisz(krazek);
 wypisz(krazek);
end.
```

Korzystając z procedur z poprzedniego programu, część główną można rozszerzyć w taki sposób, aby można było operować na kilku płytach.

```
var krazek:array[1..10] of cd;
    il,i,nr:integer;
begin
  repeat
    write('Podaj ilosc plyt
       (max. 10): ');
    readln(il);
  until (il>0) and (il<=10);
  for i:=1 to il do
    wpisz(krazek[i]);
  repeat
    repeat
      write('Ktora plyte
         wypisac (max.
         ',il,'): ');
      readln(nr);
    until (nr>=0) and
       (nr<=il);
    if (nr<>0) then
       wypisz(krazek[nr]);
  until (nr=0);
end.
```

Możliwe jest również wyszukiwanie płyty poprzez podanie nazwiska wykonawcy.

```
var krazek:array[1..10] of cd;
    il,i:integer;
    sp:string;
begin
  repeat
    write('Podaj ilosc plyt
       (max. 10): ');
    readln(il);
  until (il>0) and (il<=10);
  for i:=1 to il do
    wpisz(krazek[i]);
  write('Plyte, ktorego
     wykonawcy wypisac: ');
  readln(sp);
  for i:=1 to il do
    if (sp=krazek[i].wyk)
       then wypisz(krazek[i]);
end.
```

12 Operacje na plikach

12.1 Wpisywanie

```
program pliki;

var plik:text;
    tekst:string;

begin
  Assign(plik,'test.txt');
  Rewrite(plik);
  repeat
    readln(tekst);
    if tekst<>'' then
       writeln(plik,tekst);
  until tekst='';
  Close(plik);
end.
```

12.2 Dopisywanie

```
program pliki_dopisz;

var plik:text;
    tekst:string;

begin
Assign(plik,'test.txt');
Append(plik);
repeat
  readln(tekst);
  if tekst<>'' then
     writeln(plik,tekst);
until tekst='';
Close(plik);
end.
```

12.3 Odczytywanie

```
program pliki_czyt;

var plik:text;
    tekst:string;

begin
 Assign(plik,'test.txt');
 Reset(plik);
 repeat
  readln(plik,tekst);
  writeln(tekst);
 until EOF(plik);
 Close(plik);
end.
```

Polecenia do obsługi plików:

Assign(zmienna,plik) wiąże zmienną plikową z fizycznym plikiem

Reset(zmienna) otwiera plik związany ze zmienną do czytania

Rewrite(zmienna) otwiera plik związany ze zmienną plikową do zapisu (zamazuje zawartość, jeśli istniał)

Append(zmienna) otwiera plik związany ze zmienna plikową do zapisu (dołącza dane na końcu; tylko dla plików tekstowych)

Close(zmienna) zamyka plik związany ze zmienną plikową i zwalnia ją do ewentualnego ponownego użycia

Typy plików:

zmienna:text typ pliku tekstowego; dodatkowo można użyć poleceń: **writeln** i **readln**

zmienna:file of typ typ pliku o określonej strukturze; dane tylko określonego typu mogą być czytane i zapisywane; polecenia **reset** i **rewrite** umożliwiają zarówno pisanie jak i czytanie danych; dostępne są również polecenia: **filesize(zmienna)**, **filepos(zmienna)** oraz **seek(zmienna,pozycja)**

zmienna:file typ pliku bez określonej struktury