SILNIA

```
FUNCTION Silnia(n:INTEGER):INTEGER;
begin
   if (n=0) then silnia:=1
                else silnia:=n*silnia(n-1);
end;
```

CIĄG FIBONACZIEGO

```
FUNCTION Fibonacci(n:INTEGER):INTEGER;
BEGIN
IF n<2 THEN Fibonacci:=n
            ELSE Fibonacci:=Fibonacci(n-1)+Fibonacci(n-2)
END;
```

CIĄG FIBONACZIEGO ITERACJA

```
function FibIter(k:integer):longint;    {Wartoscia funkcji jest k-ta liczba
      Fibonacciego}
 var Fib,Fib1,Fib2:longint;
begin
 if k<2 then FibIter:=k
                else begin
                              Fib1:=0;
                              Fib2:=1;
                              while k-2>=0 do
                              begin
                                  Fib:=Fib1+Fib2;
                                  Fib1:=Fib2;
                                  Fib2:=Fib;
                                  k:=k-1
                              end;   {while }
                            FibIter:=Fib
                           end
end; {FibIter}
```

WZÓR EUCLIDESA (NWD)

```
function Euklides(a,b:integer):integer;
begin
      if a=b then Euklides:=a
             else begin
                    while a<>b do
                      if a>b then a:=a-b
                             else b:=b-a;
                    Euklides:=a;
                  end;
end; {Euklices}
```

TWORZENIE TABLICY

```
type tab=array[1..n] of integer;
var a:tab;
n - dowolna liczba dodatnia większa od 1
```

PRZESZUKIWANIE BINARNE

```
const nMax=100;
type   Tablica=Array[1..nMax] Of Integer;
Function Szukaj(Var a : Tablica; x, n : Integer) : Boolean;
Var   start, stop, sr : Integer;
Begin   start:=1;   stop:=n;
While start<stop Do
  Begin
      sr:=(start+stop) Div 2;
      If x<=a[sr] Then stop:=sr
                  Else start:=sr+1;
  End;
Szukaj:=(a[start]=x);
End;
```

PRZESZUKIWANIE BINARNE VER.2

```
const nMax=10000;
Type   Tablica = Array[1..nMax] Of Integer;

Function Szukaj(Var a:Tablica;x,start,stop:Integer):Boolean;
```

```
Var   sr:Integer;
Begin
If start=stop Then Szukaj:=(a[start]=x)
                          Else Begin
                                        sr:=(start+stop) Div 2;
                                        If x<=a[sr] Then Szukaj:=Szukaj(a,x,start,sr)
                                                          Else
        Szukaj:=Szukaj(a,x,sr+1,stop);
                          End;
End;
```

SORTOWANIE BĄBELKOWE

```
procedure   SortBubble;
begin
for i:=2 to n do
      for j:=n downto i do
           if a[j-1]>a[j] then
                begin x:=a[j-1]; a[j-1]:=a[j]; a[j]:=x end
End;
```

PORZĄDKOWANIE PRZEZ WYBÓR

```
Procedure ProsteWybieranie;
var i,min,pom:integer;
begin
       for i:=1 to n-1 do
            begin
                 min:=i; pom:=a[i]; {min indeks elementu najmniejszego, pom
      wartość}
                  {szukanie elementu najmniejszego}
                 for j:=i+1 to n do
                    if a[j]<pom then begin   min:=j; pom:=a[j] end;
                  {zamiana elementu i z elementem najmniejszym}
                 a[min]:=a[i]; a[i]:=pom;
```

```
            end
end;
```

PROSTE WSTAWIANIE

```
procedure   ProsteWstawianie;
var i,j,pom:integer;
begin
     A[0]:=-32670; {a[0] dodatkowa   skladowa dla wartownika}
     for i:=2 to n do
  begin pom:=a[i]; j:=i-1;
  {przesuwanie az do znalezienia mniejszego lub rownego elementu}
                  while pom<a[j] do
                      begin a[j+1]:=a[j]; j:=j-1; end;
          a[j+1]:=pom;
   end
end;
```

SORTOWANIE PRZEZ BINARNE WSTAWIANIE

```
procedure   BinarneWstawianie;
var i,k,L,P,pom:integer;
Begin
     for i:=2 to n do
          begin
               pom:=a[i]; L:=1; P:=i-1;
              {przeszukiwanie binarne}
               while L<=P do
                    begin k:=(L+P)div 2;
                              if pom<a[k] then P:=k-1
                                                  else L:=k+1;
                    end;
               for k:=i-1 downto L do
                    a[k+1]:=a[k];
               a[L]:=pom;
          end;
```

```
end;
```

SZYBKIE SORTOWANIE

```
Procedure QuickSort(d,g:integer);
{rekurencyjne sortowanie ciagu w x[d..g]}
Var l,p,v,pom:integer; {l – lewy element ciągu x, p - prawy}
Begin
    l:=d; p:=g;   v:=x[(d+g) div 2];
    Repeat     {podzial ciagu elementem srodkowym}
        while x[l]<v do l:=l+1;
        while v<x[p] do p:=p-1;
        if l<=p then begin {Przestaw x[l] z x[p]}
                                    pom:=x[l]; x[l]:=x[p]; x[p]:=pom; l:=l+1; p:=p-1;
                                end;
    until l>p;
    If d<p then QuickSort(d,p); {porzadkowanie podciagow}
    If l<g then QuickSort(l,g);
end; {QuickSort}
```

METODA SHELLA

```
procedure SortShella(n:word; var a:TA);
{sortowanie przez wstawianie z malejacymi przyrostami}
 var i,j,k:integer; x:OB;
begin
 k:=1; while 2*k+1<n do k:=k*2+1;   {szukanie najwiekszej odleglosci}
 while k>0 do                          {dopoki odleglosc>0                  }
  begin
    for i:=k+1 to n do
      begin
          x:=a[i]; j:=i-k;
          while (j>0) and (x<a[j]) do     {przesiewanie i-tego z odleglymi o k}
                begin a[j+k]:=a[j]; j:=j-k end;
          a[j+k]:=x
      end;
    k:=k div 2
```

```
   end
end {koniec procedury};
```

SORTOWANIE STOGOWE

```
type    OB= word;
              TA= array[1..n] of OB;
procedure SortStogowy(n:word;var a:tab);
{sortowanie drzewiaste, kopcowanie, stogowe}
 var l,p:integer; x:integer;

procedure przesianie(l,p:integer);
   var i,j:integer;
   begin
   i:=l;    j:=2*i;    x:=a[i];
   while j<=p do
     begin
     if j<p then    if a[j]<a[j+1]    then j:=j+1;
     if x>=a[j] then exit;
     a[i]:=a[j];    i:=j;    j:=2*i
     end;
   a[i]:=x;
   end {przesiania};


begin
 l:=(n div 2)+1;p:=n;
 while l>1 do
   begin l:=l-1;przesianie(l,p); end;
 while p>1 do
   begin x:=a[1];a[1]:=a[p];a[p]:=x;p:=p-1;       przesianie(1,p); end
 end {procedury heap};
```