

Rekursja

- alternatywa pętli
metoda realizacji cyklicznych operacji
- metoda na **trudne** problemy

Algorytm Euklidesa — rekursja

```
program Euklides ;  
  var m, n : integer;  
  function NWD (m,n : integer) : integer;  
  begin  
    if m <> n then  
      if m > n then NWD := NWD(m - n, n)  
      else NWD := NWD(m, n - m)  
    else  
      NWD := m  
    end;  
  begin  
    <ciało programu>  
  end.
```

Program — wywołanie funkcji NWD

```
writeln('Podaj dwie liczby naturalne');  
readln (m,n);  
if (m>0) and (n>0) then  
    writeln('NWD(',m,',',n,')= ',NWD(m, n))  
else  
    writeln('niewłaściwe dane')
```

Wieże z Hanoi

Zadanie algortymiczne — poszukiwanie przepisu na osiągnięcie celu

Przykład celu: Wieże z Hanoi.

Dane są trzy pale: **X**, **Y** i **Z**. Na jeden z nich, np. na **X** nałożono krążki o różnych średnicach, tak że krążek mniejszy nie leży pod większym. Pale **Y** i **Z** są puste. Należy przenieść wszystkie krążki z pala **X** na **Z**. Można używać **Y** jako pomocnicze miejsce przechowywania. Nie można kłaść krążków większych na mniejsze.

Wieże z Hanoi

Stan wyjściowy

oo X oo	Y	Z
ooo X ooo	Y	Z
oooo X oooo	Y	Z

Stan docelowy

X	Y	oo Z oo
X	Y	ooo Z ooo
X	Y	oooo Z oooo

Wieże z Hanoi

Stan wyjściowy

oo X oo	Y	Z
ooo X ooo	Y	Z
oooo X ooooo	Y	Z

Pierwszy ruch

X	Y	Z
ooo X ooo	Y	Z
oooo X ooooo	Y	oo Z oo

Wieże z Hanoi

Pierwszy ruch

	X		Y		Z
ooo	X	ooo	Y		Z
oooo	X	oooo	Y	oo	Z oo

Drugi ruch

	X		Y		Z
	X		Y		Z
oooo	X	oooo	ooo Y ooo		oo Z oo

Wieże z Hanoi

Drugi ruch

```
      X           Y           Z
      X           Y           Z
0000 X 0000    000 Y 000    00 Z 00
```

Trzeci ruch

```
      X           Y           Z
      X           00 Y 00    Z
0000 X 0000    000 Y 000    Z
```

Wieże z Hanoi

Trzeci ruch

	X		Y		Z
	X		oo Y oo		Z
oooo	X	oooo	ooo Y ooo		Z

Czwarty ruch

X		Y		Z
X		oo Y oo		Z
X		ooo Y ooo	oooo	Z ooooo

Wieże z Hanoi

Czwarty ruch

```

X           Y           Z
X           oo Y oo     Z
X           ooo Y ooo   ooooo Z ooooo

```

Piąty ruch

```

X           Y           Z
X           Y           Z
oo X oo     ooo Y ooo   ooooo Z ooooo

```

Wieże z Hanoi

Piąty ruch

X	Y	Z
X	Y	Z
oo X oo	ooo Y ooo	oooo Z ooooo

Szósty ruch

X	Y	Z
X	Y	ooo Z ooo
oo X oo	Y	oooo Z ooooo

Wieże z Hanoi

Szósty ruch

X	Y	Z
X	Y	000 Z 000
00 X 00	Y	0000 Z 0000

Siódmy ruch — stan docelowy

X	Y	00 Z 00
X	Y	000 Z 000
X	Y	0000 Z 0000

Wieże z Hanoi — algorytm rekursywny

```
procedure przenies (ile : integer; var Z, Na, Poprzez : kolek );  
  
begin  
  
if ile = 1 then  
    przenies_1(Z, Na)  
else  
    begin  
        przenies(ile-1, Z, Poprzez, Na);  
        przenies_1(Z, Na);  
        przenies(ile-1, Poprzez, Na, Z)  
    end  
end;  
  
end;
```

Wieże z Hanoi — algorytm rekursywny

```
procedure przenies (ile : integer; var Z, Na, Poprzez : kolek );  
  
begin  
  
if ile > 0 then  
    begin  
        przenies(ile-1, Z, Poprzez, Na);  
        przenies_1(Z, Na);  
        przenies(ile-1, Poprzez, Na, Z)  
    end  
  
end;
```

Wieże z Hanoi — algorytm rekursywny

```
program Hanoi;

const n          = 3;
type zakres      = 1..n;
type pelnyzakres = 0..n;
type kolek       = record
                    stan : array [zakres] of zakres;
                    top  : pelnyzakres;
                    name : char;
                end;
var X, Y, Z      : kolek;
```

```

procedure druk;
{ drukuje aktualne rozmieszczenie kregow na kolkach }
var i : zakres;

    procedure druk_1(linia: zakres; var k:kolek);
var i : zakres;
begin
    if k.top < linia then
    begin
        for i:= 1 to n do
            write(' ');
        write(' ',k.name,' ');
        for i:= 1 to n do
            write(' ')
        end
    end
end

```

```

else
begin
    for i:= n downto 1 do
        if k.stan[linia] < i then write(' ')
        else write('o');
    write(' ',k.name,' ');
    for i := n downto 1 do
        if k.stan[linia] < n-i+1 then write(' ')
        else write('o')
    end
end; { druk_1 }
begin
    for i:= n downto 1 do
    begin
        druk_1(i, X); write(' ');
    end
end

```

```

        druk_1(i, Y); write('      ');
        druk_1(i, Z); writeln
    end;
    writeln
end; { druk }

procedure genpusty(var K : kolek; name : char);
begin
    k.top:=0;
    k.name:=name
end; { genpusty }

procedure genpelny(var K : kolek; name : char);
var i : zakres;
begin

```

```
k.top := n;
for i:= 1 to n do
    k.stan[i] := n+1-i;
k.name:=name
end; { genpelny }
```

```
procedure przenies_1(var z,na : kolek);
begin
    na.top := na.top+1;
    na.stan[na.top] := z.stan[z.top];
    z.top:= z.top-1;
    druk
end; { przenies_1 }
```

```
procedure przenies (ile:pelnyzakres; var z, na, przez:kolek
```

```
begin
  if ile > 0 then
    begin
      przenies(ile-1, z,przez,na);
      przenies_1(z,na);
      przenies(ile-1,przez,na,z);
    end
  end; { przenies }
```

```
begin
  writeln;
  genpelny(x, 'X');
  genpusty(y, 'Y');
  genpusty(z, 'Z');
  druk;
```

```
    przenies(n,x,z,y)
end.
```

Wydruk z programu

```

  o X o           Y           Z
 oo X oo         Y           Z
ooo X ooo       Y           Z

      X           Y           Z
 oo X oo        Y           Z
ooo X ooo      Y           o Z o

      X           Y           Z
      X           Y           Z
ooo X ooo     oo Y oo       o Z o
```

X	Y	Z
X	o Y o	Z
ooo X ooo	oo Y oo	Z
X	Y	Z
X	o Y o	Z
X	oo Y oo	ooo Z ooo
X	Y	Z
X	Y	Z
o X o	oo Y oo	ooo Z ooo
X	Y	Z
X	Y	oo Z oo

o X o

Y

ooo Z ooo

X

Y

o Z o

X

Y

oo Z oo

X

Y

ooo Z ooo