# Algoritmi elementari

## Problema:

### Enunț

1. Scrieți o funcție care determină reversul unui număr dat.

Observatii:

1. reversul unui număr care se termină cu 0 va ignora cifrele 0 de la final.

### Programul

```
{
Returneaza reversul unui numar.
In:
	- n: numarul pentru care se va calcula reversul.
Out:
	- reversul lui n.
}
function revers(n:integer):integer;
var r:integer;
begin
	r:=0;
	while (n>0) do
	begin
		r:=r*10 + n mod 10;
		n:=n div 10;
	end;
	revers:=r;
end;
```

### Exemple

| Date de intrare | Rezultate |
|---|---|
| 1344 | 4431 |
| 1001 | 1001 |
| 3300 | 33 |

# Problema: palindrom

## Enunt

2. Să se verifice dacă un numar este palindrom.
a) folosind reversul
b) fara a calcula reversul (fara a parcurge tot numarul), fara sir, string sau alte structuri de date.

## Programul clasic

```
{
Determină dacă un număr dat este palindrom.
In:
       - n: numarul care se va verifica.
Out:
       - true daca n e palindrom și false dacă nu.
}
function estePalindrom1(n: integer): boolean;
begin
        estePalindrom1 := n=revers(n);
end;
```

## Programul optimizat

```
{
Determină dacă un număr dat este palindrom, intr-un mod mai eficient.
In:
       - n: numarul care se va verifica.
Out:
       - true daca n e palindrom și false dacă nu.
}
function estePalindrom2(n:integer): boolean;
var m:integer;
    sw:boolean;
begin
        sw := n mod 10 <> 0;
        m:=0;
        while (sw) and (n>0) do
        begin
                m:=m*10 + n mod 10;
                if m=n then begin sw:=true; break; end;
                n:=n div 10;
                if m=n then begin sw:=true; break; end;
                if m>n then sw:=false;
        end;
        estePalindrom2 := sw;
end;
```

## Exemple

| Date de intrare | Rezultate |
|---|---|
| 1344 | false |
| 1001 | true |
| 3300 | false |

# Problema: factorial

## Enunt

3. Determinare factorial.
 a) iterativ
 b) recursiv

## Programul iterativ

```
{
Determină factorialul unui numar. n! = 1*2*3*...*n.
In:
       - n: numarul pentru care se calculeaza factorialul.
Out:
       - factorialul lui n, n!.
}
function factorial1(n: integer):longword;
var f:longword;
   i:integer;
begin
      f:=1;
      for i:=1 to n do
            f:=f*i;
      factorial1 := f;
end;
```

## Programul recursiv

```
{
Determină factorialul unui numar. n! = 1*2*3*...*n.
In:
       - n: numarul pentru care se calculeaza factorialul.
Out:
       - factorialul lui n, n!.
}
function factorial2(n:integer):longword;
begin
        if n=0
                then factorial2:=1
                else factorial2:=n*factorial2(n-1);
end;
```

## Exemple

| Date de intrare | Rezultate |
|---|---|
| 5 | 120 |
| 0 | 1 |
| 3 | 6 |

# Problema: zerouri factorial

## Enunt

4. Fiind dat un numar natural n, sa se determine numarul de cifre 0 cu care se termina factorialul numarului n.

a) prin calcularea factorialului

b) fara calcularea factorialului.

### Programul care calculează factorialul

```
{
Determină numarul de zerouri de la sfarsitul lui n!, calculand factorialul.
In:
       - n: numarul pentru care se determina zerourile terminale in n!.
Out:
       - numarul de zerouri terminale ale lui n!.
}
function cifre0_1(n:integer):integer;
var f:longword;
    nr:integer;
begin
        f:=factorial1(n);
        nr:=0;
        while f mod 10 = 0 do
        begin
                inc(nr);
                f:=f div 10;
        end;
        cifre0_1:=nr;
end;
```

### Programul care nu calculează factorialul

```
{
Determină numarul de zerouri de la sfarsitul lui n!, fara a calcula factorialul.
In:
       - n: numarul pentru care se determina zerourile terminale in n!.
Out:
       - numarul de zerouri terminale ale lui n!.
}
function cifre0_2(n:integer):integer;
var nr,i:integer;
begin
        nr:=0;
        i:=5;
        while n div i >= 1 do
        begin
                nr:=nr + n div i;
                i:=i*5;
        end;
        cifre0_2:=nr;
end;
```

### Exemple

| Date de intrare | Rezultate |
|---|---|
| 100 | 24 |
| 5 | 1 |
| 7 | 1 |

# Problema: fibonacci

## Enunt

5. Sa se determine al n-lea termen din sirul lui Fibonacci.
a) iterativ
b) recursiv

## Programele iterative

```
{
Determină al n-lea numar fibonacci.
In:
	- n: numarul fibonacci dorit.
Out:
	- al n-lea numar fibonacci.
}
function fibo1(n:integer):integer;
var prev,crt,f,i:integer;
begin
	prev:=0; crt:=1;
	for i:=1 to n-1 do
	begin
		f:=crt+prev;
		prev:=crt;
		crt:=f;
	end;
	fibo1:=crt;
end;

{
Determină al n-lea numar fibonacci.
In:
	- n: numarul fibonacci dorit.
Out:
	- al n-lea numar fibonacci.
}
function fibo2(n:integer):integer;
var prev,crt,i:integer;
begin
	prev:=0;crt:=1;
	for i:=1 to n div 2 do
	begin
		prev:=prev+crt;
		crt:=crt+prev;
	end;
	if n mod 2 = 0
		then fibo2:=prev
		else fibo2:=crt;
end;
```

## Programele recursive

```
{
Determină al n-lea numar fibonacci.
In:
	- n: numarul fibonacci dorit.
Out:
	- al n-lea numar fibonacci.
}
function fibo3(n:integer):integer;
begin
	if (n=1) or (n=2)
		then fibo3:=1
		else fibo3:=fibo3(n-2)+fibo3(n-1);
end;
```

```
{
Determină al n-lea numar fibonacci recursiv, folosind parametri ajutatori.
In:
	- n: cate numere mai avem de calculat.
	- prev: precedentul număr fibonacci.
	- crt: numărul fibonacci curent.
Out:
	- al n-lea numar fibonacci.
}
function fibo4_aux(n,prev,crt:integer):integer;
begin
	 if (n=1) or (n=2)
		then fibo4_aux:=crt
		else fibo4_aux:=fibo4_aux(n-1, crt, prev+crt);
end;

{
Determină al n-lea numar fibonacci.
In:
	- n: numarul fibonacci dorit.
Out:
	- al n-lea numar fibonacci.
}
function fibo4(n:integer):integer;
begin
	 fibo4_aux(n, 1, 1);
end;
```

## Exemple

| Date de intrare | Rezultate |
|---|---|
| 7 | 13 |
| 5 | 5 |
| 6 | 8 |

# Problema: proprietate

## Enunt

6. Se citeste nu numar natural n. Sa se afiseze toate numerele
din intervalul [a, b) dat (a,b>0) care au propiretatea P cu n.
Doua numere au proprietatea P daca scrierile lor in baza 10 contin acelasi cifre
(ex: 23 si 3223 au proprietatea P).

## Programul

```
{
Determina o reprezentare pe biti a cifrelor unui numar.
In:
	- n: numarul dat.
Out:
	- un numar in care al k-lea bit este 1 daca exista cifra k in n.
}
function configuratie_cifre(n:integer):integer;
var b:integer;
begin
	b:=0;
	while n>0 do
	begin
		b:= b or 1 shl (n mod 10);
		n:= n div 10;
	end;
	configuratie_cifre:=b;
end;

{
Determina daca doua numere sunt in proprietatea P.
In:
	- nr1: primul numar.
	- nr2: al doilea numar.
Out:
	- true daca nr1 P nr2, false altfel.
}
function proprietateP(nr1,nr2:integer):boolean;
var b1,b2:integer;
begin
	b1:=configuratie_cifre(nr1);
	b2:=configuratie_cifre(nr2);
	proprietateP := b1=b2;
end;
{
Afiseaza toate numerele din [a, b) care sunt in proprietatea P cu n.
}
procedure afiseaza_numere(n,a,b:integer);
var i:integer;
begin
	for i:=a to b-1 do
		if proprietateP(n,i) then writeln(' ',i,' ');
end;
```

## Exemple

| Date de intrare | Rezultate |
|---|---|
| 32, 100, 1000 | 223<br>232<br>233<br>322<br>323<br>332 |