

Les algorithmes arithmétiques

I. Décomposition en facteurs premiers

La décomposition en produit de facteurs premiers, d'un entier strictement positif, consiste à écrire cet entier sous forme d'un produit de ces diviseurs premiers.

Par définition, un nombre premier ne peut pas être décomposé.

Exemples :

$$11 = 11$$

$$25 = 5 * 5$$

$$45 = 3 * 3 * 5 = 3^2 * 5$$

$$125 = 5 * 5 * 5$$

$$360 = 2 * 2 * 2 * 3 * 3 * 5$$

Principe:

+Vérifier si **n** est divisible par **2**, si oui : remplacer **n** par **n div 2** et continuer à le diviser par **2**, jusqu'à ce qu'il ne soit pas multiple de **2** ($n \text{ div } 2 \neq 0$)

+Refaire l'étape précédente pour 3, 4, ...

+Refaire les étapes précédentes jusqu'à avoir **n = 1**

Exercice :

Ecrire un programme qui stocke la décomposition en facteurs premiers d'un nombre entier strictement positif dans un tableau T puis affiche les éléments de ce tableau.

```
program facteur_premier;
uses wincrt;
type tab= array[1..10] of integer;
var n,d,nb: integer; t:tab;

procedure saisie(var n:integer);
begin
repeat
write('Entrez le nombre dont vous voulez la decomposition : ');
readln(n);
until n>0;
end;
```



```

Procedure decomposition(var t:tab;n:integer;var nb:integer);
var d:integer;
begin
  nb := 0;
  d := 2;
  While (n <> 1) do
  Begin
    if ((n mod d) = 0) then
    begin
      t[nb + 1] := d;
      n := n div d;
      nb := nb + 1;
    end
    else
      d := d + 1;
    End;
  End;
  Procedure affichage(t:tab;nb:integer);
  var i:integer;
  Begin
    for i:=1 to nb-1 do
      write(t[i], '*');
    writeln(t[nb]);
  end;

  begin
    saisie(n);
    decomposition(t,n,nb);
    affichage(t,nb);
  End.

```

II. Le calcul du factoriel

Le factoriel d'un entier n est noté par : $n! = n*(n-1)*(n-2)*...*3*2*1$

Exemple :

$$5! = 5*4*3*2*1 = 120$$

Exercice :

Ecrire un programme qui permet de :

- Saisir deux entiers strictement positifs n et p tel que $n > p$
- Calculer et afficher $A_p^n = \frac{n!}{(n-p)!}$



```
program arrange;
uses wincrt;
var n,p:integer;
procedure saisie(var n,p:integer);
begin
repeat
write('n=');read(n);
write('p=');read(p);
until (n>0) and (n>p);
end;

function fact(n:integer):longint;
var i:integer; f:longint;
begin
f:=1;
for i:=1 to n do
f:=f*i;
fact:=f;
end;

function arrangement(n,p:integer):real;
begin
arrangement:=fact(n)/fact(n-p);
end;
begin
saisie(n,p);
write('a=',arrangement(n,p):3:3);
end.
```



Exercice 1 : Les nombres parfaits

Soit n un entier naturel.

On dit que n est un nombre parfait ssi la somme de ses diviseurs vaut $2.n$.

On dit que n est un nombre parfait ssi la somme de ses diviseurs stricts vaut n .

Exemples : les premiers entiers parfaits sont 6 et 28.

Propriétés

Soit a un entier naturel.

Si a s'écrit sous la forme $2n \cdot (2n+1 - 1)$ et si $2n+1 - 1$ est premier, alors a est parfait.

On s'aperçoit ainsi que $6 = 2^1 \cdot (2^1+1 - 1)$ et $28 = 2^2 \cdot (2^2+1 - 1)$ car 3 et 7 sont premiers.

Les dérivés des nombres parfaits

Nombres "presque parfaits "

Soit n un entier naturel.

On dit que n est un nombre "presque parfait " si la somme de ses diviseurs vaut $2.n - 1$.

Toute puissance de 2 est un nombre presque parfait.

Ecrire un algorithme d'une fonction parfait qui permet de vérifier si un élément d'une matrice carrée d'entiers, M , est parfait ou non.

Solution :

0) Fonction Parfait (M : matrice; n, i, j : entier) : Booléen

1) Parfait $\leftarrow$ Faux

2) Pour k de 1 à $M[i, j]-1$ faire
 Si $M[i, j] \bmod k = 0$ Alors
 $S \leftarrow S + k$

 Fin Si

Fin Pour

3) Si $S = M[i, j]$ Alors
 Parfait $\leftarrow$ Vrai

Fin Si

4) Fin Parfait

Appeler cette fonction dans un module qui permet d'afficher tous les éléments parfaits de la matrice M .

Solution :

0) Procédure Affichage (M : matrice; n, i, j : entier)

1) Pour i de 1 à n Faire
 Pour j de 1 à n Faire
 Si Parfait (M, n, i, j) Alors
 Ecrire ($M[i, j]$)

 Fin Si

Fin Pour

Fin Pour

2) Fin Affichage



Exercice 2 :

Deux nombres **a** et **b** sont dits **nombres jumeaux** si :

- **a** et **b** sont premiers
- **a = b + 2**

Ecrire un algorithme 'une fonction **Jumeaux** permettant de tester si deux entiers **a** et **b** sont jumeaux ou non.

Exemples :

Pour **a = 7** et **b = 5**, On a : **7** et **5** sont premiers et **7 = 5 + 2**, donc sont jumeaux

Pour **a = 17** et **b = 11**, On a : **17** et **11** sont premiers mais **17 ≠ 11 + 2**, donc ne sont pas jumeaux

0) Fonction Premier (**a** : entier) : Booléen

1) **P** ← Vrai, **i** ← 2

2) Répéter

Si **a mod i = 0** Alors

P ← Faux

Sinon **i** ← **i + 1**

Fin Si

Jusqu'à (**non P OU (i=a)**)

3) Premier ← **P**

4) Fin Premier

0) Fonction jumeaux (**a**, **b**) : Booléen ;

1) Jumeaux ← Faux

Si (Premier (**a**) ET (Premier (**b**) ET (**a = b + 2**)) Alors

Jumeaux ← Vrai

Fin Si

2) Fin Jumeaux

Exercices 3 : nombres Amis

Deux nombres **a** et **b** sont dits amis si la sommes des diviseurs stricts de **a** égale à **b** et la sommes des diviseurs stricts de **b** égale à **a**. exemples : 220 et 284

Ecrire une fonction permettant de tester si deux entiers naturels sont amis ou non.



```
function amis(m,n:integer):boolean;  
Var sn,sm:integer;  
begin  
amis:=false;  
sn:=0;  
for i:=1 to n-1 do  
    if n mod i=0 then  
        sn:=sn+i;  
sm:=0;  
for i:=1 to m-1 do  
    if m mod i=0 then  
        sm:=sm+i;  
if ((sn=m) and (sm=n)) then  
    amis:=true;  
end;
```

