EMD1 d'ALGORITHMIQUE

Date : Mardi 24 Novembre 2015
Durée : 3 Heures
DOCUMENTS INTERDITS

Exercice 1 (15 pts)

Lorsque l'on prend la somme des diviseurs propres (ie. Excepté le nombre lui-même Cf. série3) d'un nombre N (>0) et que cette somme est égale à N on dit que N est un nombre parfait, mais quand cette somme est inférieure à N on dit que N est déficient, et quand elle est supérieure à N on dit que N est abondant.

Donner la **solution complète** qui nous permet de trouver :

- le plus petit nombre abondant impair (3 pts)
- et le plus petit triplet (trois éléments) de nombres abondants consécutifs (qui se suivent).(8pts)

Nota : ce triplet n'a été découvert qu'en 1975 par L. HODGES et M. REID

Mais il y a aussi ce que l'on appelle les nombres super abondants. C'est-à-dire des nombres abondants mais dont le taux d'abondance est supérieur à celui de tous les nombres abondants qui les précèdent.

Le taux d'abondance d'un nombre abondant est égal à la somme de ses diviseurs propres /n (exple : le taux d'abondance du nombre abondant 12 est égal à 1+2+3+4+6= 16/12 =1,33)

Trouver les 20 premiers nombres super abondants. (4pts)

Exercice 2 (5pts)

1. Dérouler le module pour n = 71872687 ch=7 puis pour n = 1718171216 et ch = 1
2. puis donner lui un nom et un rôle à ce module

```
Fonction CCCCCC (N,Ch: Entier):entier
variables Res,i,p,r: Entier
Fonctions nb_pos, Concat, Extr_nb
DEBUT
Res ← 0
Pour i allant de 1 à Nb_pos(N) Faire
    Dpour
    P ← Extr_Nb(N,1,i)
    Si p <> ch Alors  Res ← Concat(Res,p)
    Fpour
CCCCCC ← Res
FIN
```

Nota : on considère les fonctions suivantes déjà construites :
Fonction EXTR_NB(n : entier ; c, p : entier) : entier // Rôle : extrait de N un nombre de C chiffres à partir de la position P (incluse) et les positions sont numérotées de droite à gauche
Fonction NB_POS (n : entier) : entier // Rôle : Donne le nombre de chiffres composant N
Fonction Concat (a,b : entier) : entier // Rôle : concatène A et B

ATTENTION: Votre solution doit **ABSOLUMENT** :

- tenir compte du **formalisme et de la démarche étudiés** en cours
- être soignée (tout travail non soigné ne pourra en aucun cas prétendre à la note max. prévue dans le barème !

Alors soignez votre travail et bon courage !

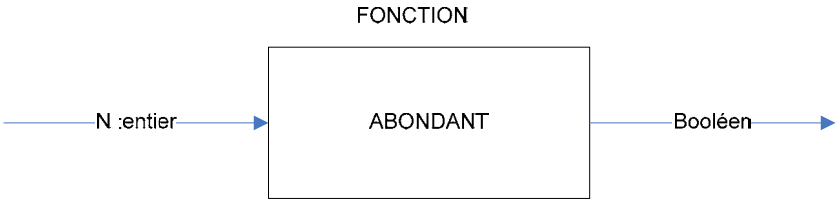
Corrigé

Nota : on aurait pu solutionner ce problème sans la modularité, cependant l'utilisation de cette dernière sera plus appréciée. Et elle sera utilisée dans la solution présentée.

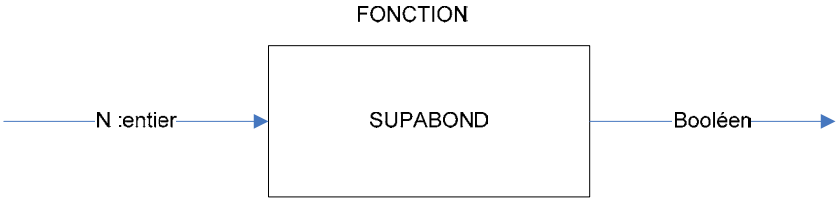
Découpage modulaire :

Le découpage est pratiquement implicite. On aura besoin :

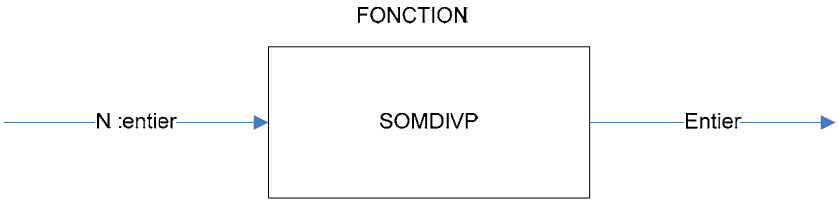
- d'un module qui vérifie si un nombre est abondant (Module ABONDANT)
- d'un module qui vérifie si un nombre est supe abondant (module SUPABOND)
- et d'un module qui nous donne la somme des diviseurs propres d'un nombre (module SOMDIVP)



Rôle : Donne Vrai si N est un nombre abondant



Rôle : Donne Vrai si N est un nombre super abondant



Rôle : Donne la somme des diviseurs propres (excepté lui-même) de N

Construction du module ABONDANT

Analyse :

- si la somme de ses diviseurs propres est supérieure à lui-même alors c'est un nombre abondant

```
fonction abondant (n: entier): entier
fonction somdivP

DEBUT
Si somdivP(n) >  n Alors abondant ← VRAI
Sinon                  abondant ← Faux
FIN
```

Construction du module SUPABOND

Analyse :

- si N n'est pas abondant le résultat (res) sera Faux
- si N est abondant
  - on calcule son taux d'abondance (Tx N= somme des diviseurs propres de N / N)
  - on met Res = Vrai // on considère que N est super abondant
  - on fait varier i = 1,2,3,4,….. // ie. On prend tous les nombres qui précèdent N
    - on calcule le taux d'abondance de i (TxI)
    - si TxI > Tx N cela signifie que l'on a trouvé un taux d'abondance supérieur à celui de N, donc N n'est pas un nombre super abondant ,car son taux d'abondance doit être le plus grand de tous ces prédécesseurs. Donc On met Res = faux

  on s'arrête si (i > N) OU (Res = faux) , on a parcouru tous les nombres < N, OU , N n'est pas super abondant

```
Fonction SupAbond (n: Entier ): Booléen
variables     i: Entier
              tx,pretx: réel
              res : booleen
Fonctions abondant , somdivP
DEBUT
Si  abondant(n) Alors
    Dsi
    I ← 1
    Res ←  Vrai
    TxN ←  somdivp(n) / n
    Répéter
    Si  abondant( i ) Alors
        Dsi
        Txi ←  somdivP(i) / i
        if  Txi > txN then  res ← Faux
        Fsi
    I ← i+1
    Jusqu'à  (i > n) OU (res =  Faux)
    SupAbond ← res
    Fsi
Sinon    supAbond ← Faux
FIN
```

Construction du module SOMDIVP

Analyse :

- on fait varier i de 1 à N/2
    - et si i est un diviseur de N on le cumule dans somme

```
FONCTION somdivP ( n : Entier) : Entier
Variables  somme, i: Entier
DEBUT
Somme ← 1
Pour i allant de 2 à n div 2 Faire
        Si n mod i=0 Alors somme ← somme+i
somdivP ← somme
FIN
```

Construction de l'algorithme principal

Analyse

On la divisera en 3 parties selon l'énoncé

Partie 1 :

- on fait varier I = 1, 2, 3,……

et on s'arrête au premier I , qui est en même temps Abondant et impair

Partie 2 :

- pour vérifier que trois nombres abondants se suivent , on prendra 3 variables AB1, AB2 , AB3 qui seront initialisées à 0
- on prend une variable booléenne (Triplet) qui nous permettra de savoir si nous avons trouvé un triplet. Elle sera initialisée à Faux
- on va faire varier i =1, 2, 3, 4,……., et à chaque fois
    - on vérifie si i est nombre abondant, si c'est le cas
        - on décale les contenus de AB1 et AB2 et on met i dans AB3, pour cela on met AB2 dans AB1, on met AB3 dans AB2 et on met I dans AB3, comme ça on aura toujours les 3 derniers nombres abondants
        - on vérifie si les trois nombres se suivent (AB2 – AB1 =1 ET AB3 – AB2 = 1) , si c'est le cas , cela signifie que l'on a trouvé un triplet , alors on met Vrai dans triplet

on s'arrêtera lorsque triplet = vrai , on a trouvé le triplet recherché

- On affiche AB1, AB2 et AB3

Partie 3:

- Donner Nb, le nombre de nombres super abondants à afficher
- On met notre compteur (NBsup) à 0
- On fait varier i =1,2,3,4,…. Et à chaque fois
    - Si i est un nombre super abondant
        - On l'affiche
        - On incrémente notre compteur Nbsup

On s'arrête lorsque NBsup = Nb , on a affiché nos NB nombres super abondants

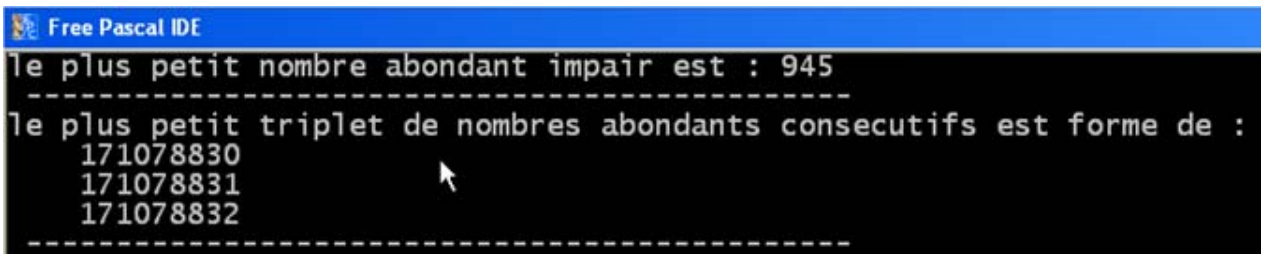
```
ALGORITHME ed1_1516
Variables i, j ,ab1 ,ab2 ,ab3 ,nbsup ,nb : entier
          tx , preTx : réel
          triplet : booléen
Fonctions abondant.fon, somdivP, Supabond

DEBUT
//--------- Question 1 ----------------------------
i ← 0
Répéter
| I ← i+1
Jusqu'à (abondant(i)) ET  (i mod 2 <>0) ;
Ecrire ('le plus petit nombre abondant impair est : ', i)
//--------- question  2 ----------------------------
AB1 ← 0
AB2 ← 0
AB3 ← 0
triplet  ← Faux
i ← 1
Répéter
|   Si  abondant(i)  Alors
|         Dsi
|         | AB1 ← AB2
|         | AB2  ← AB3
|         | AB3  ← i
|         | Si  (ab2-ab1=1) and (ab3-ab2=1)  Alors  triplet  ← Vrai
|         | Fsi
|   i ← i+1
Jusqu'à (triplet =  Vrai)
Ecrire ('le plus petit triplet de nombres abondants consecutifs est forme de : ', AB1, AB2, AB3)
//--------- Question 3 ----------------------------------------------
Ecrire (' Donner le nombre de nombres super abondants à afficher : ')
Lire (nb)
NBsup ← 0
i ← 1
répéter
| if supAbond(i)  Alors
|                       Dsi
|                       | nbsup ← nbsup +1
|                       | Ecrire (nbsup,': ',i)
|                       | Fsi
| i ← i+1
Jusqu'à  nbsup = nb
FIN
```

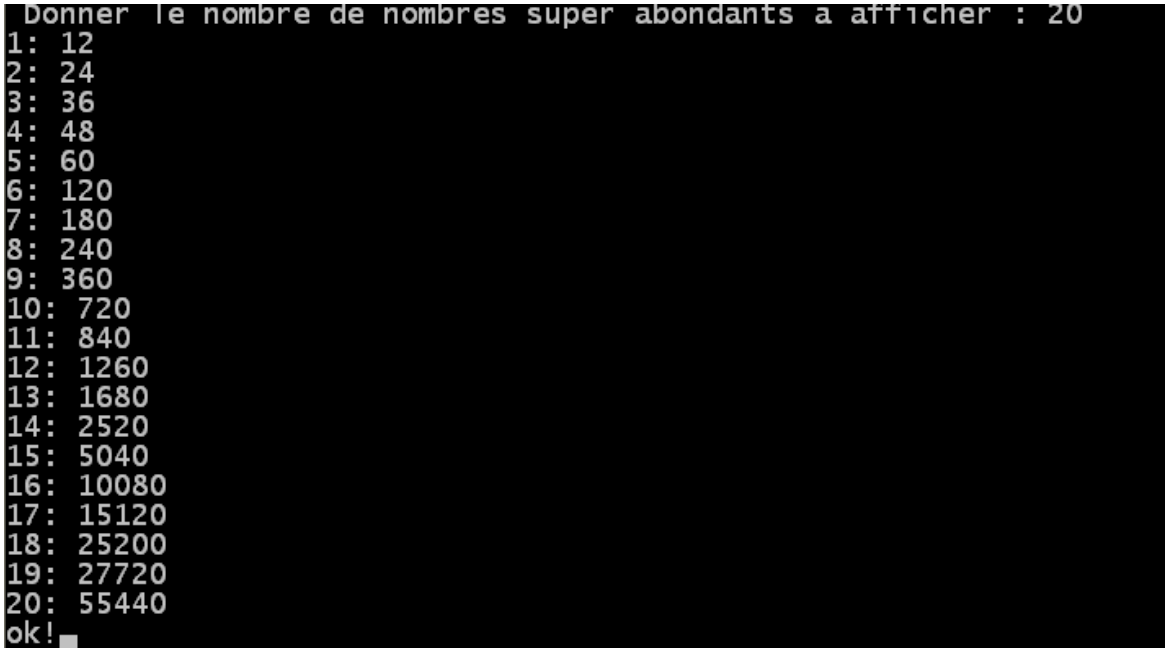
PROGRAMMATION

```
program ed1_1516;
uses crt;
var i,j,ab1,ab2,ab3,nbsup,nb:longint ;
        tx ,preTx:real;
        triplet :boolean;
{$i E:\algo\modules\abondant.fon}
{$i E:\algo\modules\somdivP.fon}
{$i E:\algo\modules\SupAbond.fon}

BEGIN
clrscr;
//--------- Question 1 -----------------------------
i:= 0;
repeat
   i:=i+1;
until (abondant(i)) and (i mod 2 <>0) ;
Writeln('le plus petit nombre abondant impair est : ',i);
writeln(' ------------------------------------------------');

//------- question  2 -----------------------------
{
AB1:=0;
AB2:=0;
AB3:=0;
triplet := false;
i:=171078800;
repeat
 if abondant(i) then
                Begin
                ab1:=ab2;
                ab2:=ab3;
                ab3:=i;
                if (ab2-ab1=1) and (ab3-ab2=1) then triplet :=true;
                end;
 i:=i+1;
until (triplet = true);
Writeln('le plus petit triplet de nombres abondants consecutifs est forme de
:');
writeln('    ', Ab1);
writeln('    ', Ab2);
writeln('    ', Ab3);
writeln(' ------------------------------------------------');     }

//--------- Question 3 ------------------------------------------------

Write(' Donner le nombre de nombres super abondants a afficher : ');
readln(nb);
NBsup:=0;
i:=1;
repeat
  if supAbond(i) then
                        Begin
                        nbsup:=nbsup +1;
                        writeln(nbsup,': ',i);
                        end;
  i:=i+1;
until nbsup =nb;
write('ok!') ;
readln;
END.
```

```
function abondant(n:longint):boolean;
// ------------------------------------
// Donne VRAI si N est un nombre abondant
// ------------------------------------
{$i E:\algo\modules\somdivP.fon}
BEGIN
if somdivP(n)>n then abondant:= true
else abondant:=false;
END;


Function SupAbond(n:longint):boolean;
// ----------------------------------------
// Donne VRAI si N est un nombre super abondant
// ----------------------------------------
var     i:integer;
        tx,pretx:real;
        res :boolean;
{$i E:\algo\modules\abondant.fon}
{$i E:\algo\modules\somdivP.fon}
BEGIN
i:=1;
res:=true;
if abondant(n) then
        Begin
        Tx:= somdivp(n)/n;
        repeat
          if abondant(i) then
                begin;
                preTx:= somdivP(i)/i;
                if  preTx > tx then  res:=false ;
                end;
         i:=i+1;
         until (i > n) OR (res = false);
        SupAbond:=res;
        end
else    supAbond:=false;
END;

FUNCTION somdivP(n:longint):longint;
(*      ------------------------------------------------------------------- *)
(*      retourne la somme des diviseurs propres de N excepte compris lui-meme
*)
(*      ------------------------------------------------------------------- *)
var somme,i:longint;
BEGIN
somme:=1;
for i:=2 to n div 2 do
            if n mod i=0 then somme:=somme+i;
somdivP:= somme;
END;
```

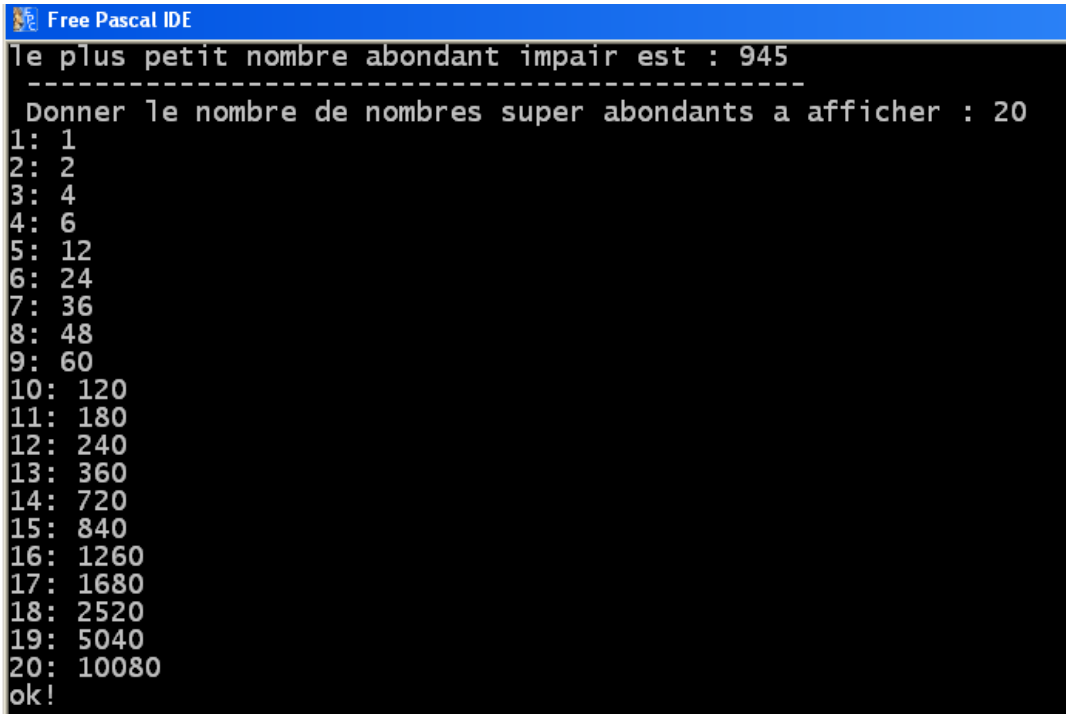
<u>ATTENTION</u> : si vous voulez rechercher le triplé formé de nombres abondants consécutifs en commençant par i = 1, je vous recommande de lancer votre programme la nuit , car il va exiger plus de 10 heures de calculs. Je comprends alors pourquoi il a fallu attendre 1975 pour le trouver. Cependant si vous voulez valider votre algorithme commencez par i = 171 000 000 par exemple. Et encore il faudra être patient car il exigera des milliards d'itérations.





AUTRE REMARQUE : en me documentant davantage sur les nombres super abondants, j'ai constaté que 1, 2, 4 et 6 sont aussi des nombres super abondants alors qu'ils ne figurent pas dans la liste fournie par la solution, car la définition la plus répandue est plutôt basée sur la somme (SOMDIV) des diviseurs (y compris le nombre lui-même). Le taux devient TxN = Somdiv (N) / N, et un nombre super abondant est un nombre dont le taux est le plus grand que tous les nombres qui le précèdent. Vous trouverez ci-après le programme du module accompagné de son résultat. J'espère qu'il est facile de reconstituer l'algorithme correspondant. Donc pour ceux qui veulent garder ce module dans leur bibliothèque, je leur recommande de garder celui-là. Il est plus simple et pourtant plus complet.

```
Function SupAbond(n:longint):boolean;
var     i:integer;
        txN,txI:real;
        res :boolean;
{$i E:\algo\modules\abondant.fon}
{$i E:\algo\modules\somdiv.fon}
BEGIN
i:=1;
res:=true;
TxN:= somdiv(n)/n;
repeat
  txi:=somdiv(i)/i;
  if  TxI > txN then  res:=false ;
  i:=i+1;
until (i > n) OR (res = false);
SupAbond:=res;
END;
```



QUESTION 2

Déroulement du 1er exemple

N	C h	Res	i	p	r	CCCCCC
71872687	**7**	0				
			1	7		
		8	2	8		
		86	3	6		
		862	4	2		
			5	7		
		8628	6	8		
		86281	7	1		
			8	7		**86281**

Déroulement du 2ième exemple

N	C h	Res	i	p	r	CCCCCC
1718171216	**1**	0				
		6	1	6		
			2	1		
		62	3	2		
			4	1		
		627	5	7		
			6	1		
		6278	7	8		
			8	1		
		62787	9	7		
			10	1		**62787**

Rôle: supprime le chiffre Ch du nombre N et inverse le résultat

Nota : bien entendu ce rôle peut être exprimé de plusieurs façons différentes. Le plus important est qu'il soit correct sur le plan sémantique.

Pour le nom : il suffit de donner une contraction des mots clés (**supprime** , **chiffre, inverse**) : SupCinv, Sup_Inv, SupChInv, S_Ch_Inv, Supinv, …… On évitera quand même de dépasser 8 caractères.

BAREME

1. Ce corrigé est un corrigé type, toute autre solution sera retenue dans la mesure où elle est cohérente et les concepts appliqués
2. Pour cet EMD La démarche peut être modulaire ou pas mais une solution modulaire sera mieux appréciée
3. Si la solution est modulaire, tenir compte , dans la notation :
   - De la Justification du découpage
   - des Dessins des modules (interfaces cohérents, rôles clairs)
4. quelle que soit la solution, tenir compte :
   - de la Structuration des idées dans l'analyse, la clarté, la précision, la concision et le soin. Et surtout, est-ce que l'analyse proposée facilite l'extraction de l'algorithme ?
5. tout travail non soigné ne pourra en aucun cas prétendre à la note max. prévue dans le barème !

EXO1	Analyse	Algorithme
PARTIE 1 (3 pts dont 1pt programmation)	• recherche d'un nombre abondant **0.5 pts** • recherche du plus petit abondant impair **0.5 pts**	Respect de l'analyse proposée, du formalisme et Présentation • **0.5** • **et 0.5 pt**
PARTIE 2 (6 pts dont 1pt programmation)	• recherche des triplets **1 pt** • test de fin de la boucle **1 pt**	Respect de l'analyse proposée , du formalisme et Présentation • **1.5** • **et 1.5 pts**
PARTIE 3 (6 pts dont 1pt programmation)	• recherche d'un nbre super abondant **1.5 pts** • recherche des Nb nombres super abondants **0.5 pts**	Respect de l'analyse proposée, du formalisme et Présentation • **2** • **et 1 pt**
Programmation sur 3 pts	**Partie 1 : 1pt** **Partie 2 : 1 pt** **Partie 3 : 1 pt** **- 0.20 pt par erreur** de programmation 15 erreurs = 0 (arrondir la note au ½ pt sup)	Tout programme qui n'aura pas un lien direct avec le problème posé et l'algorithme proposé ne sera pas pris en considération.
EXO 2	• déroulement exemple 1 : 1.5 pts (- ½ pt par erreur) • déroulement exemple 2 : 1.5 pts (- ½ pt par erreur) • rôle correct : 1 pt – simple, clair , concis, sans faute(s) • nom correct : 0.5 pt - significatif (contraction de mots clés du rôle • Ont-ils constaté que la varible R ne sert à rien ? 0.5 pt	