En fin de document le programme de l'algorithme mod 30.

Illustration de la position des 8 couples de base **> à 31.**
Si en début de programme les 8 P' ∈ [7;31] , crible le tableau → N puis on retourne à l'algorithme avec les 8 couples de Base P >31 ∈ [37;79].
1 = P ; **0** = composé, le nombre cellule =N/30, la cellule $\mathbf{A_1}$ est toujours **P' < 31,** sauf la pour la **série 1,** où A_1 = **1≠P' ;** puis on progresse de 30 : $N_{n+1} = N_n + 30$:
Illustration du positionnement des **4 couples Tbase = P'** du Gm (« groupe multiplicatif »)

0	A	B	C	E	F	G
1	**23**	1	1	1	(11*13)	1
2	(7*29)	1	1	1	(17*19)	1
3	1	0	1	$0_{=11}$	1	$0_{=13}$
4	1	1	$0_{=}$7 ; $C_{89 \geq \sqrt{N}}$	1	1	(23*31)
5	1	1	11	0 ,17	1	19
6	13	1	1	1	0	(29 ; 0
7	1	11	1	1	1	0
8	1	13	17	1	23	1
9	7 ,11, 19	1	1	1	1	1
10	31 53	0	13	1	41*43	11
11	1	17	0	1	29	1
12	1	19	1	13*7*23	11*	1
13	(37*59)	1	1	1	7 47	1
14	17	1	1	11	13	7
15	1	31	19	1	1	1
16	7		11^2*23	29		13*17
17		7		41		43
18		11	7	19	1	53*61
19	13	37 * 89 > 84		7	17	
20	11	23	31		7	
21		13	29	47*79	19	11*7*
22				17		59*67
23	7		13		11	
24	23	7	41			19
25	43		37 ; 7*17	13*31*11		
26		29		7		
27			11		53 ; 7*13	23
28	19	17				7
29	61	11	0		0	13
30	7				31	
31	29*17*11	0 ;7*19		0	23	
32	13	0	7		0	
33						
34						
35						
36				0		
37		0			0	

38 39 40	0	0	0			Lim X=7193 √ = 84 max P = 83

Le Gm, comporte une base de 8 premiers **TB_*P'>31*** appartenant à [37 **;79**] positionné en F,10 ; A,13 ; F,18 ; E,21 .
Les **0** représentent le passage des 8 **pbase** noté **pb ≤ 31** ou **Cp** (« Complémentaire premier noté **bnb dans le programme** , soit : **bnb** = (***P***' +30k) ») , qui ont marqué leurs multiples « les cellules par un 0».
Par exemple: cellule **C.25** il y a **0** ; et les deux **pb = 7** et **17**; le **0**, a été marqué par la **Tb** = **37** positionné en **A.13 ,** qui marquent leur multiples d'un **0**…
Exemple 1 :
Les bases P' peuvent être positionnées dans une même cellule « multiples » (« décomposition en facteurs premiers d'un entier 30k +23 pour cette famille…de même que plusieurs **bnb**, peuvent marquer un même multiple «*même cellule*» représenté par un **0** »)

Quelque soit N_n > 23; en partant de l'indice $n = 0$, N_n = (n * 30) + 23; **ex en: F12 = 2123 ;** il y a (71 - 1) cellules * 30 et + 23 la cellule 1 ce qui donne = **2123**
On prend le complémentaire ***: bnb ≤ √ N,*** *extrait par la base* ***Tbase P', uniquement ≤ racine carré de la limite N du crible;***
Ex :
limN = 71933; √7193 = 83,... *ce serra le plus grand* ***complémentaire Cp = bnb, extrait par la Tbase P'*** *du Gm.*

On prendra donc uniquement les complémentaires **bnb ≤ 83 = Premiers,** et ensuite les 8 P' des **Tbases** , finissent de cribler la fin du tableau fixé en marquant leur multiple d'un 0.

Exemple : Tnb=37 arrive cellule B.19 avec son complémentaire bnb = 89 > 83 ; 89 est supprimé et la TnB *fini de criblé ses multiples jusqu'au bout et sort du crible...Puis on réitère avec les Tnb suivantes en amont*

[« Mais par exemple, si on programmait avec les 8 petite bases pb ≤ 31: la pb 7 et bnb qui serait son complémentaire bnb = 59 qui était en cel B_3 est arrivé en **cellule C_4 ,** avec son complémentaire augmenté de 30 → ***bnb = 89 ≥ √ N = 83,...***
lorsque arrive le tour de cette Tbase ***7*** *et* ***bnb*** *pour cribler ; dans cette cellule les autres Tbases sont passées, son* ***bnb = 89*** *donc inutile de le tester* ***bnb%tnb = 0 ou pas*** *car il est supérieur à racine de N = 83, la* ***bnb*** *ne part pas cribler car elle est supprimée, d'où seule la Tb=7 part cribler jusqu'au bout en marquant ses multiples d'un 0 par pas de 7 puis sort du crible. On revient en amont et on continue à cribler avec les Tbases suivantes qui sont 23 et 31 positionnées en* ***G,4*** *...etc etc.»]*

Le tableau final serra sous forme de **1** = **P** ou de **0** = Composé ou produit et on fait la somme des 1

On établi un ou des tableaux virtuels carrés, de 3 000 000 sur 3 000 0000, rempli de 1 qui vont représenter les premiers à cribler, **le premier nombre** de la cellule **A.1** ≤ 31, serra toujours le premier terme de la famille à cribler. Dans l'algorithme 1 qui n'est pas un nombres premier est donc remplacer par 31 qui est une **Tbase prime** dans le programme.

Dans ce tableau en exemple ci-dessus, famille 23[30] cellule A0 = 23, puis on indexe de 30 les nombres *« cellules »* suivants, soit 30k+23 ; pour les reconstituer et uniquement ce qui seront marqué d'un 1 = premier; les 0 ne seront pas reconstitués, car ce sont des produits….

On place les 4 couples du Gm sur leur produit respectif donc dans leur cellule: ex 11*13 = 143, donc en cellule **F.1** qui est d'indice $_{n}$ = 4, elle vaut : 4*30 + 23 …etc pour les 4 autres couples de Tbase Prime du groupe multiplicatif Gm.

Déroulement :

Puis on fait démarrer le premier couple de base, **P'= 11** démarre avec son complémentaire augmenté de 30 soit C_{13} +30 ⇒ bnb = 43, et va se positionner **11** nombres plus loin, en attente dans sa cellule avec son complémentaire.

(«Comme pour Ératosthène, si ce n'est que là : les 8 Tbases P' ne vont pas jusqu'au bout et uniquement pour ces 8 Tbases P'et tant que leur complémentaire Cp bnb ≤ √ limite n , n'aura pas été extrait par leur Tbase P' , ***si bnb est > racine de N*** *alors la Tbase part marquer tous ses multiples d'un 0 et jusqu'au bout, puis sort du crible ce qui ferra accélérer le programme.»)*

Puis, c'est au tour de **13** avec son complémentaire *C_{11} augmenté de 30 bnb= 41*, va se positionner 13 nombres plus loin en attente.

On remplace le 1 par 0 dans la cellule « ***F,1 (Tbase 11.13)*** » à chaque départ d'une base P', ou le marquage d'un nombre « d'une cellule » par un complémentaire Cp =bnb qui change le 1 en 0 par pas de bnb.

Puis on arrive sur le couple de Tbase : (7 ; 29), cellule A2, on fait partir 7 avec son complément (bnb_{29} + 30 = 59), 7 nombres plus loin en attente. 29 part avec son complément (bnb_{7} + 30 = 37) ; 29 nombres plus loin on change le 1 en 0 ,….etc. Ensuite 17 avec bnb_{19} +30 puis 19 avec bnb_{17}+30 ; on change le *1en 0*, on avance…..et on arrive sur la première Tbase = 7 en attente avec son complément bnb_{59},

Le couple (7 ; 59) était arrivé sur un 1, la cellule n'est donc pas marquée 0 d'où :

59 est un possible premier, ce qui est un **1er test** de primalité pour ***bnb = 59*** , car cette cellule (7*59) n'a pas été marqué d'un 0 , par une **Tbase P'** précédente qui serait déjà passée et il n'y a pas non plus de **Tbase P'** en attente dans cette cellule **donc:**

2ème test : **Tnb =7** divise bnb=59 : Si 59%7 ≠ 0 ou bnb%Tnb ≠ 0 **alors bnb_{59} est confirmé premier** , il part cribler tous ses multiples en changeant le 1 par 0 , tous les 59 nombre, «principe d'Ératosthène» jusqu'au bout de la limite N définie puis sort du crible. «*car ce n'est pas une base P' du Gm»* .

Ensuite départ de 7, avec son nouveau bnb_{59} +30 = 89, soit bnb_{89} qui va se repositionner en attente 7 pas «cellules» plus loins…..etc.

Si le test de la division du ***bnb % Tnb*** = 59%7= 0 , alors le **C**omplémentaire de P' ***bnb*** est un ***Cc***, c'est à dire *un produit ou composé* , il est éliminé et seul Tbase **P'** repart se positionner P' pas plus loin avec son nouveau (***bnb_{59}*** + 30 = 89).

Il en serra même si une **Tbase P'**, avec son complément ***bnb*** arrive sur une cellule 0, ce qui implique que ce nombre à déjà était marqué par des complémentaires ***bnb*** > 31 qui ont été extraits et qui sont passé par ce nombre, donc l'on criblé. Alors la Tbase **P'** repart directement, **P'** nombres plus loin en attente si la cellule = 1en attente de sa prochaine itération…« *ou pas si la cellule = 0 »*, en augmentant son **bnb *de 30 à chaque saut de la Tbase P'*** , « *car cela veut dire pour chaque cellule marquée 0 que son complément* ***bnb*** *est un produit ou nombre composé ».*

(« Principe de la décomposition unique d'un produit en facteurs P »)

Ou encore : *si une Tbase P' arrive pour se positionner sur un nombre, qui est déjà occupé par une autre* ***Tbase P''****, avec son complément* ***bnb****, la aussi la Tbase P' repart P' nombres plus loin avec son (bnb +30) ; et l'autre Tbase P'' aussi avec son (bnb+30)…****puis on revient en amont,*** *pour faire partir les Tbases P' en attentes.(«ou alors on reste chaque fois en attente de la prochaine itération, mais cela ralenti le programme dans ces cas précis»)*

Etc on réitère. au fur à mesure le champs ne comporte plus que des 1 et 0 , les Tbases P' du groupe Gm sortent du crible au fur et à mesure que leur bnb > racine de N ont été extrait et sont sorti du crible et fin du criblage.

*En définitive ce Gm de Tbase noté Tnb dans le programme, agit comme un rouleau. Les **Tnb P'**, passent les une par-dessus les autres, se positionne et attendent leur tour.*
*Alors que les complémentaires **bnb premiers** < à la racine carrée du crible, eux : criblent tous leurs multiples par pas de **bnb =Prime** et change le 1 en 0 ; puis sortent directement du crible à la fin de la limite N fixée .*

***En résumé : Lorsqu'une Tnb P'**, doit partir avec son **bnb prime** ou **composé** qui est > à la racine carré du crible limite défini, alors seule la **Tnb P'** repart et crible tous ses multiples jusqu'au bout, **sans s'arrêter** puis sort du crible et on réitère avec les **Tnb** restantes…*
***Pour l'exemple cité tableau ci-dessus** avec la Tnb = 7 et bnb $_{59}$ en attente cellule C3, lorsque l'on arrive sur cette cellule le complément bnb$_{89}$ > racine carrée de N = 84,.. seul la base **Tnb = 7** fini de cribler jusqu'au bout, marque tous ses multiples d'un 0, sort du crible (fini pour lui)*
«quant à bnb$_{89}$, lui il est supprimé car > 83 il ne marquerait aucun multiple pour la limite fixée….et ainsi de suite pour les autres Bases et leur conjoint».

*Ce qui permet aux bases P' restantes qui arrive sur un 0 marqué par la plus petites base 7, de ne pas perdre de temps à faire des divisions inutiles car dans le programme **bnb%Tnb = 0** car le complémentaire est obligatoirement un produit. D'où le complémentaire est éliminé, la base P', repart avec son nouveau **bnb+30** ; pour se repositionner P' nombres plus loin; ou continue encore, si, il arrive encore sur un 0…etc.*
*À chaque saut de la base P', le Complémentaire **bnb** augmente de 30.*
*Une foi tous les Complémentaires primes > √30k+23 sont extraient par la dernière **tnb P' = 31** car c'est la plus grande, 31 fini de cribler ses multiple jusqu'au bout, sort du crible et fin du crible.*

Le programme comptabilise les 1, les transforme en premiers 30k+23 ou pas…
il peut aussi indiquer la position du dernier premier dans cette famille 30k +23.

Limite d'une famille sous Windows: environ 15 000 000 000 nombres = 30k+23, soit un criblage jusqu'à 450 000 000 000. pour la famille en question.

Exemple 2 :

Illustré par le tableau ci-dessous pour la famille 30k +1, positionnement du **Gm**, de ses 8 bases P. la première cellule en **A.1 = 31** est numéroté ou indicé $N_n = 0$
$N_n = (N_n * 30) + 31$ « *en fonction de l'indice des cellules soit par n° 0 ou n° 1* » ;

Famille 30k+1 des carrés de $P_{n\,;}$ il y aura donc 6 couples **Tnb** positionnés. De même dans la famille 19 de raison 30. les nombres premiers au carrés sont ≡ 1 ou 19 [30].

Dans la cellule E16 on a trois bases 31 *7*13 ce qui veut dire que les trois bnb sont des composés $_{c}$ 31 et bnb **91** , 7 et bnb **403** , 13 et bnb **217** ; on augmente donc les Conjoints bnb de 30 et les trois Tbases P' partent P' nombres plus loin en attente avec leur nouveau C = bnb +30...etc

Positionnement des 8 bases P, de [37 à 79] dans le programme série 1, ligne 280 :

```
void Algo::makeTab1(){
  theSerie = 1;
```

```
tBool.set(0); //veut dire cellule 0 où on met toujours le n° de la série, donc 1 pour cette série
tBool.set(3); // celulle 3 le couple (7.13) serait remplacé en position set(53) par Tbase (37,43)
tBool.set(4); // (11,11) ...remplacé  set(56); //(41,41)
tBool.set(12); // (19,19) ... set(208); //(79,79) car 49 est non prime
tBool.set(13); // (17,23) ... set(83); // (47,53)
tBool.set(28); // (29,29) ... set(116); // (59,59)
tBool.set(32) ; // (31,31) ... set(124); // (61,61)
tBase = new Base [8];
nbBase = 8;
tBase[0] = Base (7,43,10);    // (37,73,90)
tBase[1] = Base (13,37,16);   // (43,67,96)
tBase[2] = Base (11,41,15);   // (41,71,97)
tBase[3] = Base (19,79,50);   // (79,109,287)
tBase[4] = Base (17,53,30);   // (47,83,130)
tBase[5] = Base (23,47,36);   // (53,107,189)
tBase[6] = Base (29,59,57);   // (59,89,175)
tBase[7] = Base (31,61,63);   // (61,151,307)
makeTab();
}
```

0	A	B	C	E	F	G
1	P' = 31	1	(7*13)	(11*11)	1	1
2	1	1	1	(7*c_p 43)	1	19*19
3	17*23	1	(11*c_p 41)	(13*c_p 37)	7	1
4	1	1	1	1	1	7
5	1	11	1	29*29	13	(17*c_p 53)
6	0. 19et$C_{c\ 49}$	31*31	1	1	1	(23*c_p 47)
7	11	7	1	1	1	13
8	1	1	7	1	17	1
9	1	19*c79	1	7	0	1
10	1	0	(29*c_p 59)	1	7	1
11	1	1	(31*c_p 61)	1	1	7
12	1	1	1	1	1	1
13	7	1	1	1	1	1
14	1	7	1	1	0	1
15	1	1	7	1	1	1
16	1	1	1	31 *7*13	1	1
17						
18						
19						
20						
21				0 = 61^2		
22						
23						
24						
25						

26						
27						
28						
29						
30						
31						
32						
33						
34						
35						
36						
37						
38						
39						
40						Lim X=7201 √ = 84 max : C_P = 83

Ci dessous le programme éventuellement à modifier :

ligne 429 à 435:

while(nbBase>0){
for (cptBase=0 ; cptBase<nbBase ; cptBase++){
if (tBool[tBase[cptBase].ind]==0) {
tBool.set(tBase[cptBase].ind);
if (tBase[cptBase].bnb%tBase[cptBase].nb!=0) // seulement si bnb est < N limite du crible
getOut(cptBase);
}
[« *si le conjoint bnb est > à la limite N du crible alors il est supprimé et seule la Tbase tnb part cribler jusqu'à la limite N et sort du crible, puis on retourne avec les Tbases restantes et ainsi de suite… Ce qui ne semble pas avoir été pris en compte dans le programme, d'où des milliers d'opérations inutiles….* »]

Puis ligne 670 à 677 : , l'instruction n'est pas respecté il ne sauvegarde qu'un fichier de nombre celui du nombre demandé ; par exemple il a troué 3000 nombres premiers = position du dernier nombre trouvé , nombres par fichier par exemple 1000 , il devrait sauvegarder 3 fichier de 1000 nombres au lieu d'un seul… il manque probablement une instruction …

case 3:std::cout << "Entrez le nombre de nombres premiers par fichiers:" << std::endl ;
std::cout << ">" ;
std::cin >> rep ;
std::cout << "Sauvegarde des nombres premiers par fichier en cours... " ;
algo.printInFile(rep);
std::cout << "Termine!" << std::endl ;
break;

Programme C+ fait en 2003, étudiant à l'essi de Sophia Antipolis 06000
Algo P mod 30

```
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstdio>
#include <cstdlib>
#include <cstring>

#ifndef BIGTBOOL_H
#define BIGTBOOL_H
#define MAXSIZE 1000000
struct BigTBool{
  char ** tBools; //les tableaux
  long nbtab; //le nombre de tableaux
  long lastTabSize; //taille du dernier tableau
  long tailleMini; //la taille des sous tableaux
  long totalSize; //la taille totale de tous les tableaux
  BigTBool (long b, long bb = MAXSIZE);
  bool  operator [] (long in);
  void set (long in);
  ~BigTBool();
};
#endif

BigTBool::BigTBool(long b , long bb){
  b = b >> 3;
  bb = bb >> 3;
  tailleMini = bb;
  totalSize = b;
  long bi=b ;
  long cpt=1;
  bi-=bb;
  while (bi>0){
    bi-=bb;
    cpt++;
  }
  std::cout << "Nombre de tableaux : "<<(int)cpt << std::endl;
  long lasttab=0;
  if (bi!=0){
    while (bi>-bb){
      bi-=1;
      lasttab++;
    }
  }
  else{
    lasttab=bb;
  }
  std::cout << "Allocation memoire du tableau de booleen...   " << std::endl;
  if ((tBools = (char**) malloc (sizeof(char*)*cpt))==NULL){
    std::cerr << "1.Erreur d'allocation memoire dans la construction des tableaux du  BigTBool." <<
std::endl;
```

```
    exit(1);
  }
  long i;
  for (i=0 ; i<cpt ; i++){

    if ((tBools[i] = (char*) calloc (bb,sizeof(char)))==NULL){
      std::cerr << "2.Erreur d'allocation memoire dans la construction des tableaux du  BigTBool." <<
std::endl;
      exit(1);
    }
    std::cout << "Allocation du tableau" << (int) i+1 << "/" << (int)cpt <<std::endl;
  }
  if ((tBools[cpt-1] = (char*) calloc (lasttab,sizeof(char)))==NULL){
    std::cerr << "3.Erreur d'allocation memoire dans la construction des tableaux du  BigTBool." <<
std::endl;
    exit(1);
  }
  std::cout << "Termine!" << std::endl;
  nbtab = cpt;
  lastTabSize = lasttab;
}

bool BigTBool::operator [] (long in){
  int reste = in & 0x0000000000000007;
  in = in >> 3;
  long cpttab=0;
  while (in>tailleMini){
    in-=tailleMini;
    cpttab++;
  }
  char ourChar = tBools[cpttab][in] ;
  switch(reste){
  case 0:
    if ((ourChar & 0x80) == 0x00)
      return false;
    else
      return true;
    break;
  case 1:
    if ((ourChar & 0x40) == 0x00)
      return false;
    else
      return true;
    break;
  case 2:
    if ((ourChar & 0x20) == 0x00)
      return false;
    else
      return true;
    break;
  case 3:
    if ((ourChar & 0x10) == 0x00)
      return false;
```

```
    else
      return true ;
    break;
  case 4:
    if ((ourChar & 0x08) == 0x00)
      return false;
    else
      return true;
    break;
  case 5:
    if ((ourChar & 0x04) == 0x00)
      return false;
    else
      return true;
    break;
  case 6:
    if ((ourChar & 0x02) == 0x00)
      return false;
    else
      return true;
    break;
  case 7:
    if ((ourChar & 0x01) == 0x00)
      return false;
    else
      return true;
    break;
  default:
    return false;
  }
}

void BigTBool::set (long in){
  int reste = in & in & 0x0000000000000007;
  in = in >> 3;
  long cpttab=0;
  while (in>tailleMini){
    in-=tailleMini;
    cpttab++;
  }
  char & refb = tBools[cpttab][in];
  switch(reste){
  case 0:
    refb =refb | 0x80;
    break;
  case 1:
    refb =refb | 0x40;
    break;
  case 2:
    refb =refb | 0x20;
    break;
  case 3:
    refb =refb | 0x10;
```

```
    break;
  case 4:
    refb =refb | 0x08;
    break;
  case 5 :
    refb =refb | 0x04;
    break;
  case 6 :
    refb =refb | 0x02;
    break;
  case 7:
    refb =refb | 0x01;
    break;
  }
}

BigTBool::~BigTBool(){
  free(tBools);
}

struct Base{
  int nb;
  long bnb;
  long ind;
  bool stop;
  Base();
  Base(int n, long bn, long i);
  void next();
};

Base::Base() : bnb(0){
}

Base::Base(int n, long bn , long i) : bnb(bn){
  nb=n;
  ind=i;
}

void Base::next(){
  ind+=nb;
  bnb+=30;

}

#ifndef ALGO_H
#define ALGO_H
#define LIMITE 18446744073709551615
#define NBMAXBOOL 200
#define NBLIGNEMAX 1000000

#define NBCASE 500000

struct Algo{
```

```
  BigTBool tBool;
  long nbCase;
  Base * tBase;
  int nbBase;
  int theSerie;
  void getOut(int cpt);
  void MAJtBase();
  Algo();
  Algo(long taille);
  Algo(long taille , long tailleb);
  void makeTab();
  void makeTab1();
  void makeTab7();
  void makeTab11();
  void makeTab17();
  void makeTab19();
  void makeTab23();
  void makeTab29();
  void makeTab13();
  void printInFile(long nb);
  void showLast();
  void showNieme(long limit);
  char * int64ToChar(long nb);
  friend std::ostream & operator << (std::ostream & os , Algo & b);
};
#endif

long charToInt64 (char str [255]){
  long result =0;
  long coeff =1;
  int lg = strlen(str);
  int ind = lg-1;
  while (ind>=0){
    result+=coeff*(((int)str[ind])-48);
    ind--;
    if (coeff==1)
      coeff=10;
    else
      coeff*=10;
  }
  return result;
}

Algo::Algo () : nbCase(10000) , tBool(10000){
}

Algo::Algo(long taille):  nbCase(taille) , tBool(taille){
}

Algo::Algo(long taille, long tailleb):  nbCase(taille) , tBool(taille,tailleb){
}

void Algo::makeTab13(){
```

```
  theSerie = 13;   // lancer en debut d'Algo les 8 bases de 7 a 31 a partir de leur cellule set( ),jusqu'au bout et marquent leur mulitples
  tBool.set(0);     // puis retour au programme avec les nouvelles Tbase de 37 a 79 dans leur cellule set()et on lance l'Algo
  tBool.set(4);  // (7.19) ... set(97)(37,79)
  tBool.set(8);  // (11.23)... set(72)(41,53)
  tBool.set(13);  // (13.31) ... set(87)(43,61)
  tBool.set(16);  // (17.29)... set(92)(47,59)
  tBase = new Base [8];
  nbBase = 8;
  tBase[0] = Base (7,79,18);   // (37,109,134)
  tBase[1] = Base (19,37,23);  // (79,67,176)
  tBase[2] = Base (11,53,19);  // (41,83,113)
  tBase[3] = Base (23,41,31);  // (53,71,125)
  tBase[4] = Base (13,61,26);  // (43,151,216) // ici bnb 91 et 121 non prime donc tBase 43 et bnb 151 cellule 216
  tBase[5] = Base (31,43,44);  // (61,73,148)
  tBase[6] = Base (17,59,33);  // (47,89,139)
  tBase[7] = Base (29,47,45);  // (59,107,210) // idem 77 non prime donc tBase 59 et bnb 107 cellule 210
  makeTab();
}

void Algo::makeTab1(){
  theSerie = 1;
  tBool.set(0); //veut dire cellule 0 on met le n° de la serie donc 1
  tBool.set(3); // celulle 3 le couple (7.13) qui serait remplacé en position set(53) par Tbase (37,43)
  tBool.set(4); // (11,11) ...remplacé  set(56); //(41,41)
  tBool.set(12); // (19,19) ... set(208); //(79,79) car 49 est non prime
  tBool.set(13); // (17,23) ... set(83); // (47,53)
  tBool.set(28); // (29,29) ... set(116); // (59,59)
  tBool.set(32) ; // (31,31) ... set(124); // (61,61)
  tBase = new Base [8];
  nbBase = 8;
  tBase[0] = Base (7,43,10);    // (37,73,90)
  tBase[1] = Base (13,37,16);   // (43,67,96)
  tBase[2] = Base (11,41,15);   // (41,71,97)
  tBase[3] = Base (19,79,50);   // (79,109,287)
  tBase[4] = Base (17,53,30);   // (47,83,130)
  tBase[5] = Base (23,47,36);   // (53,107,189)
  tBase[6] = Base (29,59,57);   // (59,89,175)
  tBase[7] = Base (31,61,63);   // (61,151,307)
  makeTab();
}

void Algo::makeTab7(){
  theSerie = 7;
  tBool.set(0);
  tBool.set(6); // (11.17) ou cellule set(64) (41.47)
  tBool.set(7); // (7.31)  ou cellule set(75) (37.61)
  tBool.set(8);  // (13.19) ou cellule set(113) (43.79)
  tBool.set(22); // (23.29) ou cellule set(104) (53.59)
  tBase = new Base [8];
```

```
  nbBase = 8;
  tBase[0] = Base (11,47,17);  // (41,107,146)
  tBase[1] = Base (17,41,23);  // (47,71,111)
  tBase[2] = Base (7,61,14);  // (37,151,186)
  tBase[3] = Base (31,37,38);  // (61,67,136)
  tBase[4] = Base (13,79,34);  // (43,109,156)
  tBase[5] = Base (19,43,27);  // (79,73,192)
  tBase[6] = Base (23,59,45);  // (53,89,157)
  tBase[7] = Base (29,53,51);  // (59,83,163)
  makeTab();
}

void Algo::makeTab11(){
  theSerie = 11;
  tBool.set(0);
  tBool.set(5);  // (7.23) ou cellule set(64) (37.53)
  tBool.set(7);  // (13.17) ou cellule set(64) (43.47)
  tBool.set(11); // (11.31) ou cellule set(64) (41.61)
  tBool.set(18); // (19.29) ou cellule set(64) (59.79.)
  tBase = new Base [8];
  nbBase = 8;
  tBase[0] = Base (7,53,12);   // (37,83,102)
  tBase[1] = Base (23,37,28);  // (53,67,118)
  tBase[2] = Base (13,47,20);  // (43,107,153)
  tBase[3] = Base (17,43,24);  // (47,73,114)
  tBase[4] = Base (11,61,22);  // (41,151,206)
  tBase[5] = Base (31,41,42);  // (61,71,144)
  tBase[6] = Base (19,59,37);  // (59,109,214)
  tBase[7] = Base (29,79,76);  // (79,89,234)
  makeTab();
}

void Algo::makeTab17(){
  theSerie = 17;
  tBool.set(0);
  tBool.set(2);   // (7.11) ou set(50)(37.41)
  tBool.set(12);  // (13.29) ou set(84)(43.59)
  tBool.set(14);  // (19.23) ou set(139)(79.53)
  tBool.set(17);  // (17.31) ou set(95)(47.61)
  tBase = new Base [8];
  nbBase = 8;
  tBase[0] = Base (7,41,9);    // (37.71,87)
  tBase[1] = Base (11,37,13);  // (41.67,91)
  tBase[2] = Base (13,59,25);  // (43.89,127)
  tBase[3] = Base (29,43,41);  // (59.73,143)
  tBase[4] = Base (19,53,33);  // (79.83,218)
  tBase[5] = Base (23,79,60);  // (53.109,192)
  tBase[6] = Base (17,61,34);  // (47.151,236)
  tBase[7] = Base (31,47,48);  // (61.107,217)
  makeTab();
}

void Algo::makeTab19(){
```

```
  theSerie = 19;
  tBool.set(0);
  tBool.set(1);  // (7.7) ou set(45) (37.37)
  tBool.set(5);  // (13.13) ou set(61) (43.43)
  tBool.set(9);  // (17.17) ou set(73) (47.47)
  tBool.set(10); // (11.29) ou set(80) (41.59)
  tBool.set(17); // (23.23) ou set(93) (53.53)
  tBool.set(19); // (19.31) ou set(81) (31.79)
  tBase = new Base [8];
  nbBase = 8;
  tBase[0] = Base (7,37,8);    // (37.67,82)
  tBase[1] = Base (13,43,18);  // (43.73,104)
  tBase[2] = Base (17,47,26);  // (47.107,167)
  tBase[3] = Base (11,59,21);  // (41.89,121)
  tBase[4] = Base (29,41,39);  // (59.71,139)
  tBase[5] = Base (23,53,40);  // (53.83,146)
  tBase[6] = Base (19,61,38);  // (31.109,112)
  tBase[7] = Base (31,79,81);  // (79.61,160)
  makeTab();
}

void Algo::makeTab23(){
  theSerie = 23;
  tBool.set(0);
  tBool.set(4);  // (11.13) ou set(58) (41.43)
  tBool.set(6);  // (7.29)  ou set(72) (37.59)
  tBool.set(10); // (17.19) ou set(123) (47.79)
  tBool.set(23); // (23.31) ou set(107) (53.61)
  tBase = new Base [8];
  nbBase = 8;
  tBase[0] = Base (11,43,15);  // (41.73,99)
  tBase[1] = Base (13,41,17);  // (43.71,101)
  tBase[2] = Base (7,59,13);   // (37.89,109)
  tBase[3] = Base (29,37,35);  // (59.67,131)
  tBase[4] = Base (17,79,44);  // (47.109,170)
  tBase[5] = Base (19,47,29);  // (79.107,281)
  tBase[6] = Base (23,61,46);  // (53.151,266)
  tBase[7] = Base (31,53,54);  // (61.83,168)
  makeTab();
}

void Algo::makeTab29(){
  theSerie = 29;
  tBool.set(0);
  tBool.set(3);  // (7.17) ou set(57) (37.47
  tBool.set(6);  // (11.19) ou set(107) (41.79)
  tBool.set(9);  // (13.23) ou set(75) (43.53)
  tBool.set(29); // (29.31) ou set(119) (59.61)
  tBase = new Base [8];
  nbBase = 8;
  tBase[0] = Base (7,47,10);   // (37.107,131)
  tBase[1] = Base (17,37,20);  // (47.67,104)
  tBase[2] = Base (11,79,28);  // (41.109,148)
```

```
  tBase[3] = Base (19,41,25);  // (79.71,186)
  tBase[4] = Base (13,53,22);  // (43.83,118)
  tBase[5] = Base (23,43,32);  // (53.73,128)
  tBase[6] = Base (29,61,58);  // (59.151,296)
  tBase[7] = Base (31,59,60);  // (61.89,180)
  makeTab();
}

void Algo::makeTab(){
  int cptBase;
  int pourcent =0;
  long nbpourcent =0;
  long nbdix = nbCase/10;
  std::cout << pourcent << "% effectue..." << std::endl ;
  while(nbBase>0){
    for (cptBase=0 ; cptBase<nbBase ; cptBase++){
      if (tBool[tBase[cptBase].ind]==0) {
            tBool.set(tBase[cptBase].ind);
            if (tBase[cptBase].bnb%tBase[cptBase].nb!=0)
              getOut(cptBase);
      }
      tBase[cptBase].next();
      if (tBase[cptBase].ind>nbCase)
            tBase[cptBase].stop=1;

    }
    if (tBase[0].ind>nbpourcent+nbdix){
      nbpourcent=tBase[0].ind;
      pourcent+=10;
      std::cout << pourcent << "% effectue..." << std::endl ;
    }
    MAJtBase();
  }
}

void Algo::MAJtBase(){
  int i;
  for (i=0 ; i<nbBase ; i++){
    if (tBase[i].stop==1){
      int j;
      for (j=i ; j<nbBase-1 ; j++){
            tBase[j].bnb=tBase[j+1].bnb;
            tBase[j].nb=tBase[j+1].nb;
            tBase[j].ind=tBase[j+1].ind;
            tBase[j].stop=tBase[j+1].stop;
      }
      nbBase--;
    }
  }
}

std::ostream & operator << (std::ostream & os , Algo & b){
  long i =0;
```

```
  long n=13;
  os << (int)n << std::endl;
  for (i=0 ;  i<b.nbBase ; i++){
    if (!b.tBool[i])
      os << (int)n << std::endl;
    n+=30;
  }
  return os;
}

void Algo::printInFile(long nb){
  long i=0;
  long v=1;
  long n=theSerie;
  FILE * fic;
  long cptfic = 1;
  char nomficinit [10];
  sprintf(nomficinit,"serie%d",theSerie);
  char  nomfic [80];
  sprintf(nomfic,"%s-%dl",nomficinit,cptfic);
  bool enable = true;
  char * strn;
  char * strv;
  if ((fic=fopen(nomfic,"w"))==NULL)
    std::cerr << "erreur durant l'ouverture du fichier " <<nomfic ;
  for (i=1 ; i<nbCase; i++){
    n+=30;
    if (!tBool[i]){
      enable = true ;
      v++;
      strn = int64ToChar(n);
      strv = int64ToChar(v);
      fprintf(fic,"%s ",strn);
      fprintf(fic,"%s\n",strv);
      delete(strn);
      delete(strv);
    }
    if ((v%nb==0) && (enable)){
      enable=false;
      fclose(fic);
      cptfic++;
      sprintf(nomfic,"%s-%dl",nomficinit,cptfic);
      fclose(fic);
      if ((fic=fopen(nomfic,"w"))==NULL)
            std::cerr << "erreur durant l'ouverture du fichier serie" << theSerie;
    }
  }
  fclose(fic);
}

void Algo::showLast(){
  long i= 0 ;
  long v= 0;
```

```
  long n=theSerie;
  long realn=n;
  n=theSerie+((nbCase-1)*30);
  for (i=nbCase ; v<1; i--){
    if (!tBool[i]){
      realn=n;
      v++;
    }
    n-=30;
  }
  std::cout <<"le nombre: " << int64ToChar(realn)<< "    ";
  v=0 ;
  for (i=0 ; i<nbCase; i++){
    if (!tBool[i]){
      v++;
    }
  }
  std::cout << "position: "<< int64ToChar(v) << std::endl ;
}

void Algo::showNieme(long limit){
  long i= 0;
  long v= 0;
  long n=theSerie;
  long realn=n;
  for (i=0 ; (i<nbCase) && (v<limit); i++){
    if (!tBool[i]){
      v++;
      realn=n;
    }
    n+=30;
  }
  std::cout << "position: "<< int64ToChar(v) << "    "<<"le nombre: " << int64ToChar(realn)<<
std::endl ;
}

void Algo::getOut(int cpt){
  long indtemp = tBase[cpt].ind;
  while (nbCase-indtemp>tBase[cpt].bnb){
    indtemp+=tBase[cpt].bnb;
    tBool.set(indtemp);
  }
}

char * Algo::int64ToChar(long nb){
  char * result;
  long coeff =1;
  int lg = 1;
  while (nb/(coeff*10)!=0){
    lg++ ;
    if (coeff==1)
      coeff=10;
    else
```

```
    coeff*=10 ;
  }
  result = new char [lg+1];
  int i;
  for (i=0 ; i<lg ; i++){
   result[i]=(nb/coeff)+48;
   nb-=(nb/coeff)*coeff;
   coeff/=10;
  }
  result[lg]='\0';
  return result;
}

int main (int argc, char * argv []){
  int rep ;
  char strrep [255];
  long tailleTab;
  long tailleTBool;
  std::cout << "Entrez le nombre de case du tableau a construire:" << std::endl;
  std::cout << ">";
  std::cin >> strrep;
  tailleTab=charToInt64(strrep);
  std::cout << "Entrez le nombre de case des sous-tableaux:" << std::endl;
  std::cout << ">";
  std::cin >> strrep;
  tailleTBool=charToInt64(strrep);
  Algo algo (tailleTab,tailleTBool);
  std::cout << "Choisissez votre serie:" << std::endl;
  std::cout << "   1: 1" << std::endl;
  std::cout << "   2: 7" << std::endl;
  std::cout << "   3: 11" << std::endl;
  std::cout << "   4: 13" << std::endl;
  std::cout << "   5: 17" << std::endl;
  std::cout << "   6: 19" << std::endl;
  std::cout << "   7: 23" << std::endl;
  std::cout << "   8: 29" << std::endl;
  std::cout << ">";
  std::cin >> rep;
  switch (rep) {
  case 1:
   std::cout << "Construction du tableau pour la serie 1...   " <<std::endl;
   algo.makeTab1();
   break;
  case 2:
   std::cout << "Construction du tableau pour la serie 7...   " <<std::endl;
   algo.makeTab7();
   break;
  case 3:
   std::cout << "Construction du tableau pour la serie 11...   "<<std::endl;
   algo.makeTab11();
   break;
  case 4:
   std::cout << "Construction du tableau pour la serie 13...   " <<std::endl;
```

```
  algo.makeTab13();
  break;
case 5:
  std::cout << "Construction du tableau pour la serie 17...   "<<std::endl;
  algo.makeTab17();
  break;
case 6:
  std::cout << "Construction du tableau pour la serie 19...   "<<std::endl;
  algo.makeTab19();
  break;
case 7:
  std::cout << "Construction du tableau pour la serie 23...   " <<std::endl;
  algo.makeTab23();
  break;
case 8:
  std::cout << "Construction du tableau pour la serie 29...   " <<std::endl;
  algo.makeTab29();
  break;
default:
  std::cout << "erreur - Appuyer sur CTRL-C pour quitter le programme" << std::endl;
}
std::cout << "Termine!" << std::endl;
while (1){
  std::cout << "Quelle action voulez-vous effectuer:" << std::endl;
  std::cout << "   1. Afficher le dernier nombre trouve" << std::endl;
  std::cout << "   2. Afficher le n-ieme nombre" << std::endl ;
  std::cout << "   3. Enregistrer tous les nombres premiers trouves par fichiers" << std::endl;
  std::cout << "   4. Quitter le programme" << std::endl ;
  std::cout << ">";
  std::cin >> rep;
  switch (rep) {
  case 1:
    std::cout << "Affichage du dernier nombre trouve..." << std::endl;
    algo.showLast();
    std::cout << std::endl;
    break;
  case 2:
    std::cout << "Entrez la position du nombre premier:" <<std::endl;
    std::cout << ">";
    std::cin >> rep;
    std::cout << "Affichage du " << rep << " ieme nombre trouve..." << std::endl;
    algo.showNieme(rep);
    std::cout << std::endl;
    break;
  case 3:
    std::cout << "Entrez le nombre de nombres premiers par fichiers:" << std::endl;
    std::cout << ">";
    std::cin >> rep;
    std::cout << "Sauvegarde des nombres premiers par fichier en cours...  ";
    algo.printInFile(rep);
    std::cout << "Termine!" << std::endl ;
    break;
  case 4:
```

```
      exit(1);
      break;
    default:
      std::cout << "erreur - Appuyer sur CTRL-C pour quitter le programme" << std::endl;
    }
  }
  return 0;
}
```