Programme fait en 2003, il y a une petite imperfection , ne pas tenir compte de l'édition du dernier nombre
et dans le déroulement du programme le programmeur n'a pas tenu compte de la limite des nombres premiers
« conjoints » < à racine de N extraient par le groupe des 8 nombres premier de base et qui vont marquer leur multiples
jusqu'à la limite N fixée,
Ce qui ne gène en rien sur le nombres de premiers extraient mais qui n'optimise pas l'algorithme….
Fonctionne avec code::block

```
#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 ;
  tBool.set(0) ;
  tBool.set(4) ;
  tBool.set(8) ;
  tBool.set(13);
  tBool.set(16) ;
  tBase = new Base [8] ;
  nbBase = 8 ;
  tBase[0] = Base (7,49,11);
  tBase[1] = Base (19,37,23);
  tBase[2] = Base (23,41,31);
  tBase[3] = Base (11,53,19);
  tBase[4] = Base (13,61,26);
  tBase[5] = Base (31,43,44);
  tBase[6] = Base (17,59,33);
  tBase[7] = Base (29,47,45);
  makeTab();
}

void Algo::makeTab1(){
  theSerie = 1 ;
  tBool.set(0) ;
  tBool.set(3) ;
  tBool.set(4) ;
  tBool.set(12) ;
  tBool.set(13);
  tBool.set(28);
  tBool.set(32) ;
  tBase = new Base [8] ;
  nbBase = 8 ;
  tBase[0] = Base (7,43,10);
  tBase[1] = Base (13,37,16);
  tBase[2] = Base (11,41,15);
```

```
  tBase[3] = Base (19,49,31);
  tBase[4] = Base (17,53,30);
  tBase[5] = Base (23,47,36);
  tBase[6] = Base (29,59,57);
  tBase[7] = Base (31,61,63);
  makeTab();
}

void Algo::makeTab7(){
  theSerie = 7;
  tBool.set(0);
  tBool.set(6) ;
  tBool.set(7) ;
  tBool.set(8);
  tBool.set(22) ;
  tBase = new Base [8] ;
  nbBase = 8 ;
  tBase[0] = Base (11,47,17);
  tBase[1] = Base (17,41,23);
  tBase[2] = Base (7,61,14);
  tBase[3] = Base (31,37,38);
  tBase[4] = Base (13,49,21);
  tBase[5] = Base (19,43,27);
  tBase[6] = Base (23,59,45);
  tBase[7] = Base (29,53,51);
  makeTab();
}

void Algo::makeTab11(){
  theSerie = 11;
  tBool.set(0);
  tBool.set(5);
  tBool.set(7);
  tBool.set(11) ;
  tBool.set(18);
  tBase = new Base [8] ;
  nbBase = 8 ;
  tBase[0] = Base (7,53,12);
  tBase[1] = Base (23,37,28);
  tBase[2] = Base (13,47,20);
  tBase[3] = Base (17,43,24);
  tBase[4] = Base (11,61,22);
  tBase[5] = Base (31,41,42);
  tBase[6] = Base (19,59,37);
  tBase[7] = Base (29,49,47);
  makeTab();
}

void Algo::makeTab17(){
  theSerie = 17;
  tBool.set(0);
  tBool.set(2);
  tBool.set(12) ;
  tBool.set(14);
  tBool.set(17);
  tBase = new Base [8] ;
  nbBase = 8 ;
  tBase[0] = Base (7,41,9);
  tBase[1] = Base (11,37,13);
  tBase[2] = Base (13,59,25);
```

```
  tBase[3] = Base (29,43,41);
  tBase[4] = Base (19,53,33);
  tBase[5] = Base (23,49,37);
  tBase[6] = Base (17,61,34);
  tBase[7] = Base (31,47,48);
  makeTab();
}

void Algo::makeTab19(){
  theSerie = 19;
  tBool.set(0);
  tBool.set(1);
  tBool.set(5);
  tBool.set(9);
  tBool.set(10);
  tBool.set(17);
  tBool.set(19);
  tBase = new Base [8] ;
  nbBase = 8 ;
  tBase[0] = Base (7,37,8);
  tBase[1] = Base (13,43,18);
  tBase[2] = Base (17,47,26);
  tBase[3] = Base (11,59,21);
  tBase[4] = Base (29,41,39);
  tBase[5] = Base (23,53,40);
  tBase[6] = Base (19,61,38);
  tBase[7] = Base (31,49,50);
  makeTab();
}

void Algo::makeTab23(){
  theSerie = 23;
  tBool.set(0) ;
  tBool.set(4);
  tBool.set(6);
  tBool.set(10) ;
  tBool.set(23);
  tBase = new Base [8] ;
  nbBase = 8 ;
  tBase[0] = Base (11,43,15);
  tBase[1] = Base (13,41,17);
  tBase[2] = Base (7,59,13);
  tBase[3] = Base (29,37,35);
  tBase[4] = Base (17,49,27);
  tBase[5] = Base (19,47,29);
  tBase[6] = Base (23,61,46);
  tBase[7] = Base (31,53,54);
  makeTab();
}

void Algo::makeTab29(){
  theSerie = 29;
  tBool.set(0) ;
  tBool.set(3);
  tBool.set(6);
  tBool.set(9);
  tBool.set(29) ;
  tBase = new Base [8] ;
  nbBase = 8 ;
  tBase[0] = Base (7,47,10);
```

```
  tBase[1] = Base (17,37,20);
  tBase[2] = Base (11,49,17);
  tBase[3] = Base (19,41,25);
  tBase[4] = Base (13,53,22);
  tBase[5] = Base (23,43,32);
  tBase[6] = Base (29,61,58);
  tBase[7] = Base (31,59,60);
  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 trouves dans des 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 nombre premier par fichier:" << std::endl ;
      std::cout << ">" ;
      std::cin >> rep ;
      std::cout << "Sauvegarde des nombres premiers 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 ;
}
```