```
/*
-------------------------------------------------------------------------------
Nom du fichier  : main.cpp
Nom du labo     : Labo 5 : reflex clavier
Auteur(s)       : Bouatit Nikola et Métrailler Jérémie
Date creation   : 19.11.2021
Description :
                  Ce programme permet de générer et d'afficher la liste des nombres
                  premiers compris de 1 à une valeur saisie par l'utilisateur
                  Ensuite il y aura deux affichage le premier consistant a montrer
                  la taille des tableaux générée avec 10 colonnes
                  Deuxièmement le programme affiche de nouveau le tableau mais en
                  ayant marquée les cases avec un charactere au emplacement ou se
                  trouve les numéros premiers
                  Finalement il y aura un texte affichant le nombre de chiffre
                  premier trouver puis une liste de ces derniers
                  Le programme ensuite se termine

Remarque(s) :


Compilateur : Mingw-w64 g++ 11.2.0
-------------------------------------------------------------------------------
*/
#include <cstdlib>   // Nécéssaire pour le EXIT_SUCCESS
#include <iostream>  // Nécéssaire pour l'affichage dans la console
#include <limits>    // Nécéssaire pour vidage du buffer
#include "crible.h"
#include "affichageTableau.h"
#include "saisieUtilisateur.h"


using namespace std;
int main() {
    // Début de programme
    cout << "Bienvenue" << endl;


   const int      LIM_INFERIEUR = 2;
   const int      LIM_SUPERIEUR = 100;
   const int      NB_COLONNE    = 10;
   const string   MSG_SAISIE    = "Veuiller rentrer le nombre de valeur a generer ";
   const string   MSG_ERREUR    = "Erreur veuillez saisir une valeur comprise dans les "
                                  "limites ";

   const unsigned TAILLE_TABLEAU = (unsigned)saisieControlee( LIM_INFERIEUR,
                                                              LIM_SUPERIEUR,
                                                              MSG_SAISIE,
                                                              MSG_ERREUR);

   int tabNbPremiers[TAILLE_TABLEAU];
   unsigned int taille = 0 ;

    //Criblage du tableau
   cribler( tabNbPremiers,
            taille,
            TAILLE_TABLEAU);

   //Affichage
   cout << "initialisation du tableau" << endl;
   afficherTableau(TAILLE_TABLEAU, NB_COLONNE);
   cout << endl<< "Criblage du tableau" << endl;
   afficherTableau(tabNbPremiers,TAILLE_TABLEAU,NB_COLONNE,'X');
   cout << "il y a "<< taille << " nombres premiers qui sont : ";
   ecrireTableau(taille,tabNbPremiers);

    // fin de programme
    cout << "\nPresser ENTER pour quitter";
```

```
    cin.ignore(numeric_limits<streamsize>::max(), '\n');
 return EXIT_SUCCESS;
}
```

```
/*
------------------------------------------------------------------------------
Nom du fichier : saisieUtilisateur.h
Nom du labo    : HEIGVD-PRG1-CRIBLE
Auteur(s)      : Jérémie Métrailler, Nikola Bouattit
Date creation  : 19.11.2021

Description    : Fonctions de saisie contrôlée. La saisie contrôlée
                 s'assure que la saisie corresponde exactement à ce qui est
                 attendu par le code appelant.

Remarque(s)    : Supplément d'informations :
                 - Contrôle l'intégrité du buffer d'entrée après la saisie.
                 - L'affichage des bornes est ajouté au message de saisie.

Assertions     : (borneInf <= borneSup) uniquement saisies non- binaires :
                   Problème : La borne supérieure est plus grande que la borne
                              inférieure.
                   Solution : Contrôler l'ordre des bornes dans l'appel de la
                              fonction.
                 (borneGauche <= borneDroite) uniquement saisies non-binaires :
                   Problème : La borne droite est plus grande que la borne gauche.
                   Solution : Contrôler l'ordre des bornes dans l'appel de la
                              fonction.

Modifications  : n/a

Compilateur    : Mingw-w64 g++ 11.2.0
------------------------------------------------------------------------------
*/

#ifndef SAISIE_UTILISATEUR_H
#define SAISIE_UTILISATEUR_H

#include <string>

/**
 * @Nom                 : saisieControlee
 * @But                 : S'assure que la saisie soit dans les bornes (bornes
 *                        comprises) ou qu'elle corresponde à l'une des bornes et
 *                        qu'elle contienne uniquement ce qui est attendu.
 * @param borneInf      : Borne inférieure.
 * @param borneSup      : Borne supérieure.
 * @param MESSAGE_SAISIE : Message d'invite pour la saisie (l'affichage des bornes
 *                        se fait dans la fonction).
 * @param MESSAGE_ERREUR : Message à afficher en cas d'erreur.
 * @return              : Saisie contrôlée.
 */
int saisieControlee(int borneInf, int borneSup,
                    const std::string& MESSAGE_SAISIE,
                    const std::string& MESSAGE_ERREUR);

#endif
```

```
/*
-----------------------------------------------------------------------------
Nom du fichier : saisieUtilisateur.cpp
Nom du labo    : HEIGVD-PRG1-CRIBLE
Auteur(s)      : Jérémie Métrailler, Nikola Bouattit
Date creation  : 19.11.2021

Description    : Fonction de saisie contrôlée. La saisie contrôlée
                 s'assure que la saisie corresponde exactement à ce qui est
                 attendu par le code appelant.

Remarque(s)    : Saisie contrôlée :
                 - Contrôle l'intégrité du buffer d'entrée après la saisie.
                 - Contrôle que la valeur soit dans les bornes (bornes comprises)
                 - Contrôle que le buffer ne contienne uniquement la saisie.
                 - L'affichage des bornes est ajouté au message de saisie.
                 - Si la borne inférieure ou gauche est plus grande que la borne
                   supérieure ou droite. Quitte le programme et affiche une
                   erreur d'assertion.

Modifications  : n/a

Compilateur    : Mingw-w64 g++ 11.2.0
-----------------------------------------------------------------------------
*/
#include <iostream>
#include <limits>
#include <string>
#include <cassert>
#include "saisieUtilisateur.h"
using namespace std;

int saisieControlee(int borneInf, int borneSup,
                    const string& MESSAGE_SAISIE,
                    const string& MESSAGE_ERREUR) {

   // Contrôle la cohérence des bornes
   assert(borneInf <= borneSup);

   bool erreur;
   int saisie;
   do {
      // Affiche le message de saisie ainsi que les bornes
      cout << MESSAGE_SAISIE << " [" << borneInf << ".." << borneSup << "] : ";
      cin  >> saisie;

      // Contrôle que la valeur soit dans les bornes
      erreur = cin.fail() or saisie <  borneInf or saisie >  borneSup;

      if (erreur) {
         cout << MESSAGE_ERREUR << endl;
         cin.clear();
      }
      cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Vider le buffer

   } while (erreur);

   return saisie;
}
```

```
/*
-----------------------------------------------------------------------------
Nom du fichier : affichageTableau.h
Nom du labo    : HEIGVD-PRG1-CRIBLE
Auteur(s)      : Jérémie Métrailler, Nikola Bouattit
Date creation  : 19.11.2021

Description    : Librairie permettant d'afficher les contenue d'un tableau

Remarque(s)    : Cette librairie a été spécifiquement péparer pour la lecture de
                 tableau contenant des valeurs numériques comme des chiffre premiers

Assertions     : n/a

Modifications  : n/a

Compilateur    : Mingw-w64 g++ 11.2.0
-----------------------------------------------------------------------------
*/

#ifndef HEIGVD_PRG1_CRIBLE_AFFICHAGETABLEAU_H
#define HEIGVD_PRG1_CRIBLE_AFFICHAGETABLEAU_H

/**
 * @Nom                 : afficherTableau
 * @But                 : Affiche une matrice ou pour chaque valeur du tableau fournis le signale
 par un
 *                        charactere spécial
 * @param tab           : Tableau a fournir
 * @param taille        : Taille du tableau
 * @param nbColonne     : Nombre de colonne pour l'affichage
 * @param charSpecia    :  Charactere spéciale pour signaler une valeur égale au tableau
 */
void afficherTableau(const int tab[], unsigned taille, unsigned nbColonne = 10,
                     char charSpecial = 'X');
/**
 * @Nom                 : afficherTableau
 * @But                 : Affiche une matrice d'une certaine taille et colonne
 * @param taille        : Taille de la matrice a générée
 * @param nbColonne     : nombre de colonne (par défaut 10)
 */
void afficherTableau(unsigned taille, unsigned nbColonne=10);
/**
 * @Nom                 : ecrireTableau
 * @But                 : Affiche tous les élément contenue dans un tableau
 * @param tab           : Tableau de donnée a lire
 * @param taille        : Nombre d'élément actuellement dans le tableau
 */
void ecrireTableau(unsigned taille, const int tab[]);



#endif //HEIGVD_PRG1_CRIBLE_AFFICHAGETABLEAU_H
```

```
/*
-----------------------------------------------------------------------------
Nom du fichier : affichageTableau.cpp
Nom du labo    : HEIGVD-PRG1-CRIBLE
Auteur(s)      : Jérémie Métrailler, Nikola Bouattit
Date creation  : 19.11.2021

Description    : Les fonctions permettant l'affichage de la matrice basé sur un
                 tableau 1 dimension

Remarque(s)    : Seulement les tableaux a une seule dimension sont supporté
                 Les matriche affiche pour chaque ligne le nombre maximal de
                 colonne

Assertions     : n/a

Modifications  : n/a

Compilateur    : Mingw-w64 g++ 11.2.0
-----------------------------------------------------------------------------
*/

#include "affichageTableau.h"
#include <iostream>     //Nécessaire pour l'écriture dans la console
#include <iomanip>      //Nécessaire pour limiter l'affichage


const char REPRESENTANT    = '0';   // Charactere pour l'affichage de la matrice
const int ESPACE_AFFICHAGE =  3;    // Espace définie pour un affichage constant

using namespace std;


void afficherTableau(const int tab[], unsigned taille, unsigned nbColonne,
                     char charSpecial){

   unsigned caseTableau = 0;
   if(nbColonne == 0)
   {
      for (unsigned ligne = 1; ligne <= taille; ++ligne) {
         if (ligne == (unsigned)tab[caseTableau]) {
            cout  << fixed << left
                  << setw(ESPACE_AFFICHAGE) << REPRESENTANT;
            ++caseTableau;
         } else {
            cout  << fixed <<left
                  << setw(ESPACE_AFFICHAGE) << charSpecial;
         }
      }
      cout << endl;
   }
   else {
      for (unsigned ligne = 1; ligne <= taille;) {
         for (unsigned colonne = 0; colonne < nbColonne; ++colonne) {
            if (ligne == (unsigned)tab[caseTableau]) {
               cout  << fixed << left
                     << setw(ESPACE_AFFICHAGE) << REPRESENTANT;
               ++caseTableau;
            } else {
               cout  << fixed << left
                     << setw(ESPACE_AFFICHAGE) << charSpecial;
            }
            ++ligne;
         }
         cout << endl;
      }
   }
}
```

```
void afficherTableau( unsigned taille, unsigned nbColonne){
   if(nbColonne == 0){
      for(unsigned i =0; i < taille;++i){
         cout  << fixed << left
               << setw(ESPACE_AFFICHAGE) << REPRESENTANT;
      }
   }
   else {
      for (unsigned ligne = 0; ligne < taille; ++ligne) {
         for (unsigned colonne = 0; colonne < nbColonne; ++colonne) {
            cout  << fixed << left
                  << setw(ESPACE_AFFICHAGE) << REPRESENTANT;
            ++ligne;
         }
         cout << endl;
      }
   }
}

void ecrireTableau(unsigned taille, const int tab[]) {
   for (unsigned int i = 0; i < taille; ++i) {
      cout << fixed << left << setw(ESPACE_AFFICHAGE) << tab[i];
   }
}
```

```
/*
-----------------------------------------------------------------------------
Nom du fichier : crible.h
Nom du labo    : HEIGVD-PRG1-CRIBLE
Auteur(s)      : Jérémie Métrailler, Nikola Bouattit
Date creation  : 19.11.2021

Description    : Librairie permettant àl'utilisateur d'utiliser la méthode du crible
                 d'Eratosthène afin de trouver les nombres premiers
                 jusqu'au nombre qu'il aura spécifié

Remarque(s)    : Cette librairie ne met à disposition qu'une seule fonction afin
                 de simplifier au plus la tâche de l'utilisateur

Assertions     : n/a

Modifications  : n/a

Compilateur    : Mingw-w64 g++ 11.2.0
-----------------------------------------------------------------------------
*/

#ifndef HEIGVD_PRG1_CRIBLE_CRIBLE_H
#define HEIGVD_PRG1_CRIBLE_CRIBLE_H

/**
 * @Nom                   : cribler
 * @But                   : Remplir un tableau de nombre consécutif en començant
 *                          par le chiffre 2 puis éléminier chaque nombre
 *                          qui ne sont pas des nombre premiers jusqu'au nombre
 *                          limite passé en paramètre
 * @param tabNbPremeiers  : Tableau de nombres premiers
 * @param taille          : Taille du tableau de nombres premiers
 * @param nbLimite        : Nombre jusqu'au quel il faut trouver les nombre premiers
 *
 */
void cribler(int tabNbPremiers[], unsigned& taille, unsigned nbLimite);

#endif
```

```
/*
-----------------------------------------------------------------------------
Nom du fichier : crible.cpp
Nom du labo    : HEIGVD-PRG1-CRIBLE
Auteur(s)      : Jérémie Métrailler, Nikola Bouattit
Date creation  : 19.11.2021

Description    : Corps de la librairie crible. Il exisite dans le corps de cette librairie des
fonctions qui ne sont
                 pas mise à disposition de l'utilisateur comme par exemple remplirTableau,
                 chercherMutliple et
                 supprimerMultiple.

Remarque(s)    : Les fonctions listées précédemment ne sont pas mise à disposition de l'utilisateur
car elles sont
                 spécifiques à cette librairie et cela n'aurait pas de sens et compliquerait la tâche
                 de l'utilisateur
                 de les lui faire utiliser

Assertions     : n/a

Modifications  : n/a

Compilateur    : Mingw-w64 g++ 11.2.0
-----------------------------------------------------------------------------
*/
#include "crible.h"

/**
 * @Nom             : remplirTableau
 * @But             : Remplit un tableau de nombre consécutifs en commençant par 2
 * @param tab       : Tableau à remplir
 * @param taille    : Taille du tableau à remplir
 */
void remplirTableau(int tab[], unsigned taille);

/**
 * @Nom             : chercherMultiple
 * @But             : Rechercher le multiple du diviseur passé en paramètre
 * @param tab       : Tableau dans lequel chercher
 * @param taille    : Taille du tableau
 * @param diviseur  : Diviseur dont il faut chercher les multiples
 * @param pos       : Position du multiple
 * @return          : La position du multiple
 */
unsigned chercherMultiple(const int tab[], unsigned taille, int diviseur,
                          unsigned pos = 0);

/**
 * @Nom             : supprimerMultiple
 * @But             : Supprime tous les multiples d'un diviseur passé en paramètre
 *                    contenu dans un tableau
 * @param tab       : Tableau à parcourir
 * @param taille    : Taille du tableau
 * @param pos       : Position de l'élément à supprimer
 */
void supprimerMultiple(int tab[], unsigned& taille, unsigned pos);

void cribler(int tabNbPremiers[], unsigned& taille, unsigned nbLimite) {
   // La capacité du tableau est égal au nombre limite - 1 car on commence à compter
   // à partir de 1 mais le tableau sera rempli à partir du nombre 2 car 1 est une
   // exception dans les nombre premiers
   const unsigned CAPACITE = nbLimite - 1;
   taille = CAPACITE; // le tableau est plein
   remplirTableau(tabNbPremiers, CAPACITE);

   unsigned pos = 0;
   for (unsigned diviseur = 2; diviseur <= taille; ++diviseur) {
```

```
      while ( (pos = chercherMultiple(tabNbPremiers, taille, (signed)diviseur, pos))
               != taille ) {
         supprimerMultiple(tabNbPremiers, taille, pos);
         ++pos; // chercher à partir de la position suivant celle trouvée
      }
      pos = 0; // réinitialisation de la position
   }
}

void remplirTableau(int tab[], unsigned taille) {
   for (unsigned i = 0; i < taille; ++i) {
      tab[i] = (int)i + 2 ; // On remplit le tableau en commençant par 2
   }
}


unsigned chercherMultiple(const int tab[], unsigned taille, int diviseur,
                          unsigned pos) {

   for ( ; pos < taille; ++pos) {
      if (diviseur != tab[pos] and tab[pos] % diviseur == 0) {
         return pos;
      }
   }
   return taille;

}

void supprimerMultiple(int tab[], unsigned& taille, unsigned pos) {
   if (pos < taille) {
      for (unsigned i = pos + 1; i < taille; ++i) {
         tab[i - 1] = tab[i];
      }
      --taille;
   }
}
```