```
1   /*
2   ------------------------------------------------------------------------------
3   Nom du fichier    : main.cpp
4   Nom du labo       : Labo08_Galton Groupe L
5   Auteur(s)         : Jeremiah Steiner, Ylli Fazlija
6   Date creation     : 11.01.2022
7   Description (But) : Le but du programme est d'utiliser la classe Galton afin de
8                           définir la répartiion des billes (stoqué dans un vecteur de int)
9                           et de l'afficher sous forme graphique afin d'afficher une courbe
10                          Gaussienne.
11  Remarque(s)       : Les bornes min et max sont modifiables dans les
12                          valeurs positives des int.
13  Compilateur       : Mingw-w64 g++ 8.1.0
14  ------------------------------------------------------------------------------
15  */
16
17  #include <iostream>         // Input-Output
18  #include <cstdlib>          // EXIT_SUCCESS
19  #include <limits>           // Vider le buffer
20  #include "Galton.h"         // Classe Galton permettant de créer des instances de
21                              // planches.
22  #include "Utilitaires.h"    // Librairie de fonctions génériques, utiles
23  #include "Dictionnaire.h"   // Fichier d'en-tête qui stocke
24                              // nos caractères et chaînes de caractères.
25
26  #define VIDER_BUFFER() std::cin.ignore(std::numeric_limits<streamsize>::max(), '\n')
27
28  using namespace std;
29
30  int main() {
31
32     const unsigned  BORNE_MIN                    =    1;
33     const unsigned  BORNE_MAX_BILLE           = 1000;
34     const unsigned  BORNE_MAX_HAUTEUR           =  100;
35
36      // message d'accueil
37      Afficher(MESSAGE_DEMARRAGE, true);
38
39      bool recommencer;
40      int nbrDeBille;
41      int hauteur;
42
43      do {
44          // Stocker les entrées utilisateur
45          nbrDeBille  = LireUnNombre(BORNE_MIN,
46                                     BORNE_MAX_BILLE,
47                                              MESSAGE_PROMPT_BILLES,
48                                     MESSAGE_ERREUR_NOMBRE,
49                                              true);
50
51          hauteur     = LireUnNombre(BORNE_MIN,
52                                     BORNE_MAX_HAUTEUR,
53                                     MESSAGE_PROMPT_HAUTEUR,
54                                     MESSAGE_ERREUR_NOMBRE,
55                                     true);
56
57          // Création de l'objet + affichage
58          Galton g((unsigned)nbrDeBille,
59                 (unsigned)hauteur);
60
61        // permet d'afficher le tableu de int en ligne
62          // cout << g.getTableauBilles();
63          // cout << endl;
64
65          g.AfficherTableauGraphique(CARACTERE_AFFICHAGE,
66                                     CARACTERE_ESPACEMENT);
67
68          //  Lire l'entrée utilisateur de redémarrage
69          recommencer = (estCharDansVect(
70                    LireChar(MESSAGE_RECOMMENCER,
71                                CARACTERES_AFFICHAGE_FIN,
72                                true),
73                                CARACTERES_VALIDATION_FIN));
74
75      } while (recommencer);
76
```

```
77        Afficher(MESSAGE_FIN, false);
78        VIDER_BUFFER();
79
80       return EXIT_SUCCESS;
81    }
82
83
```

```
///*
-------------------------------------------------------------------------------
Nom du fichier    : Galton.h
Nom du labo       : Labo08_Galton Groupe L
Auteur(s)         : Jeremiah Steiner, Ylli Fazlija
Date creation     : 11.01.2022
Description (But) : Fichier d'en-tête de la classe Galton. Chaque instance de
                    cette classe représente un plateau avec une hauteur
                    et un nombre de billes donné.
Remarque(s)       :
Compilateur       : Mingw-w64 g++ 8.1.0
-------------------------------------------------------------------------------
*/

#ifndef LABO08_GALTON_GALTON_H
#define LABO08_GALTON_GALTON_H

#include <vector>
#include <ostream>   // Permets d'afficher un tableau en utilisant un output stream.
#include <random>

/// Déclaration de la classe Galton
/// Chaque instance contient un nombre de billes et une hauteur.
/// Les propriétés nbrDeBilles et hauteur sont constantes.
/// Ceci est un choix car lorsque l'objet est instancié, on ne souhaite pas
/// qu'il soit modifié ensuite.
class Galton {

   // Propriétés de la planche
    const unsigned nbrDeBilles;
    const unsigned hauteur;
    std::vector<int> tableauBilles; // TODO : mettre en unsigned ?
    static std::random_device rd;  // Sera utilisé comme seed pour l'aléatoire.

public:
   /// Constructeur par défaut
   /// On ne l'utilisera pas.
   Galton()=delete;

   /// Constructeur par copie
   /// \param g
   Galton(Galton& g);

   /// Constructeur paramétrique
   /// \param nbBille Nombre de billes à lancer dans la planche.
   /// \param h Hauteur de la planche.
    Galton(unsigned nbBille, unsigned h);

   /// Getter simple pour obtenir le tableau de billes d'une instance.
   /// \return Tableau de billes de l'instance.
   std::vector<int> getTableauBilles() const;

   /// Permets d'afficher le tableau de billes sous forme graphique
   /// \param CARACTERE Caractère qui représentera une bille.
   /// \param ESPACE Caractère qui représentera un espace.
    void AfficherTableauGraphique(const char& CARACTERE, const char& ESPACE) const;

private:
   /// Permets de lancer les billes dans la planche et remplir le tableau de billes.
   void LancerBilles();
};


#endif //LABO08_GALTON_GALTON_H

```

```cpp
/*
-----------------------------------------------------------------------------------
Nom du fichier    : Galton.cpp
Nom du labo       : Labo08_Galton Groupe L
Auteur(s)         : Jeremiah Steiner, Ylli Fazlija
Date creation     : 11.01.2022
Description (But) : Définition de la classe Galton. Contient les différentes
                    fonctions utiles à l'utilisation d'un plateau.
Remarque(s)       :
Compilateur       : Mingw-w64 g++ 8.1.0
-----------------------------------------------------------------------------------
*/


#include <iostream>
#include <algorithm>  // Utilisé pour trouver le plus grand élément dans un tableau.
#include "Galton.h"

using namespace std;

// Création de la seed pour le random.
std::random_device Galton::rd;

/// Constructeur paramétrique de l'objet Galton
/// \param nbBille le nombre de billes à lancer dans la planche
/// \param h la hauteur de la planche
Galton::Galton(const unsigned nbBille, const unsigned h) :
   nbrDeBilles(nbBille),
   hauteur(h)
{
   tableauBilles.resize(h + 1); // on s'assure d'avoir la place

   // Remplissage du tableau de billes.
   LancerBilles();
}

/// Constructeur par copie de la classe Galton
/// Il crée une nouvelle instance à partir des valeurs d'une autre instance et
/// lance une nouvelle fois la simulation. Les résulats de celle-ci ne seront
/// donc pas les mêmes.
/// \param g instance à copier.
Galton::Galton(Galton& g):
   nbrDeBilles(g.nbrDeBilles),
   hauteur(g.hauteur)
{
   tableauBilles.resize(g.hauteur + 1); // on s'assure d'avoir la place
   LancerBilles();
}

/// Cette fonction permet de remplir le tableau de billes des valeurs
/// On lance chaque bille et on détermine, en utilisant un générateur de
/// chiffres aléatoire, son emplacement.
void Galton::LancerBilles()
{
   std::mt19937 gen(rd()); // Moteur d'aléatoire qui utilise le seed rd.

   // On règle notre moteur d'aléatoire
   // pour nous donner des chiffres aléatoires entre 0 et 1.
    std::uniform_int_distribution<> distrib(0, 1);

    size_t indice;

   // Boucle qui itère sur le nombre de billes.
   // Chaque itèration correspond à une bille.
    for(size_t i = 0; i < nbrDeBilles; ++i) {
        indice = 0;

      // Deuxième boucle qui itère sur la hauteur.
      // à chaque itération, deux issues sont possibles : droite ou gauche.
      // Si distrib retourne 1, on ajoute 1 à indice.
        for (size_t n = 0; n < hauteur; ++n) {
            indice += distrib(gen);
        }

      // Une fois la deuxième boucle terminée, on a trouvé l'indice de
      // la case dans laquelle la bille tombe.
```

```
77          tableauBilles[indice]++;
78      }
79  }
80
81  /// Permets d'afficher une représentation graphique du tableau
82  /// à l'aide d'une double boucle.
83  /// \param CARACTERE Le caractère qui représentera une bille.
84  /// \param ESPACE Le caractère qui représentera les espaces.
85  void Galton::AfficherTableauGraphique(const char& CARACTERE,
86                                        const char& ESPACE) const
87  {
88      for (size_t i = ((size_t) *max_element(tableauBilles.begin(),
89                                             tableauBilles.end())); i > 0; --i)
90      {
91          for (size_t val : tableauBilles)
92          {
93              cout << (val >= i ? CARACTERE : ESPACE);
94          }
95          cout << endl;
96      }
97  }
98
99  /// Simple getter afin d'accéder au tableau de billes d'une instante Galton.
100 vector<int> Galton::getTableauBilles() const {
101    return tableauBilles;
102 }
103
```

```
/*
------------------------------------------------------------------------------
Nom du fichier    : Utilitaires.h
Nom du labo       : Labo08_Galton Groupe L
Auteur(s)         : Jeremiah Steiner, Ylli Fazlija
Date creation     : 11.01.2022
Description (But) : Fichier d'en-tête de la librairie utilitaire.
Remarque(s)       :
Compilateur       : Mingw-w64 g++ 8.1.0
------------------------------------------------------------------------------
*/

#ifndef LABO08_GALTON_UTILITAIRES_H
#define LABO08_GALTON_UTILITAIRES_H

#include <vector> // Surcharge de l'opérateur de flux afin d'afficher un vecteur.

/// Lis un nombre entré par l'utilisateur, et retourne ledit nombre
/// \param borneMin Valeur minimale acceptable (inclue)
/// \param borneMax Valeur maximale acceptable (inclue)
/// \param msgPrompt Message de demande
/// \param msgErreur Message à Afficher en cas d'erreur
/// \param afficherBornes Est-ce que les bornes sont affichées?
/// \remark Vérification des bornes: [borneMin; borneMax]
/// \return Le nombe récupéré du flux
int LireUnNombre(int borneMin,
                 int borneMax,
                 const std::string &msgPrompt,
                 const std::string &msgErreur,
                 bool afficherBornes);

/// Permet de lire un character, après avoir afficher un message et
/// une liste de char en rapport avec la question, et de le renvoyer
/// \param msgPrompt une string d'affichage
/// \param affichageChars les valeurs prises en compte
/// \param afficherCharsVal boolean, true les char d'information sont affichés
/// \return le charactere saisi
char LireChar(const std::string& msgPrompt,
              const std::vector<char>& AffichageChars,
              bool afficherCharsVal);

/// Affiche un message dans la console. Fonction surchargée
/// \param message Message à Afficher
/// \param retourLigne Est-ce qu'il faut faire un retour de ligne?
void Afficher(const std::string& message,
              bool retourLigne = true);

/// Affiche un caractère dans la console. Fonction surchargée
/// \param caractere Caractère à Afficher
/// \param retourLigne Est-ce qu'il faut faire un retour de ligne?
void Afficher(char caractere,
              bool retourLigne = true);

/// Surcharge de l'opérateur de flux afin d'afficher un vecteur.
/// \param os Flux de sortie
/// \param v Vecteur à afficher.
/// \return Référence au flux de sortie.
std::ostream& operator <<(std::ostream& os,
                          const std::vector<int>& v);

/// Permet de savoir si un char apparait dans une vecteur de char
/// \param c char a tester
/// \param listDeChar un vecteur de char
/// \return true si le char est dans la liste, false sinon
bool estCharDansVect(const char& c,
                     const std::vector<char>& listDeChar);

#endif //LABO08_GALTON_UTILITAIRES_H
```

```cpp
/*
-----------------------------------------------------------------------------
Nom du fichier    : Utilitaires.cpp
Nom du labo       : Labo08_Galton Groupe L
Auteur(s)         : Jeremiah Steiner, Ylli Fazlija
Date creation     : 11.01.2022
Description (But) : Fichier de définition de la librairie Utilitaires. Permets des
                    entrées et sorties basiques.
Remarque(s)       :
Compilateur       : Mingw-w64 g++ 8.1.0
-----------------------------------------------------------------------------
*/

#include <string>
#include <limits>          // Vider le buffer
#include <iostream>
#include "Dictionnaire.h"
#include "Utilitaires.h"

#define VIDER_BUFFER() std::cin.ignore(std::numeric_limits<streamsize>::max(), '\n')

using namespace std;

/// Lis un nombre entré par l'utilisateur, et retourne ledit nombre
/// \param borneMin Valeur minimale acceptable (inclue)
/// \param borneMax Valeur maximale acceptable (inclue)
/// \param msgPrompt Message de demande
/// \param msgErreur Message à Afficher en cas d'erreur
/// \param afficherBornes Est-ce que les bornes sont affichées?
/// \remark Vérification des bornes: [borneMin; borneMax]
/// \return Le nombe récupéré du flux
int LireUnNombre(int borneMin,
                     int borneMax,
                     const string& msgPrompt,
                     const string& msgErreur,
                     bool afficherBornes)
{
    bool erreur;
    int nombreLu;

    do {
        Afficher(msgPrompt, false);

        // Afficher les bornes seulement si on a besoin
        if (afficherBornes)
        {
            Afficher(CARACTERE_ESPACE, false);
            Afficher("["s + to_string(borneMin) + " - " + to_string(borneMax) + "]", false);
        }

        // Afficher fin de prompt et lire nombre
        Afficher(CARACTERE_FIN_PROMPT, false);
        Afficher(CARACTERE_ESPACE, false);
        cin >> nombreLu;

        // Vérification d'erreurs
        erreur = cin.fail() || nombreLu < borneMin || nombreLu > borneMax;
        cin.clear();
        VIDER_BUFFER();

        // La méthode Afficher retourne void, donc on ne peut pas l'utiliser dans la clause
        // du while pour afficher un message en cas d'erreur
        if(erreur)
        {
            Afficher(msgErreur);
        }

    } while(erreur);

    return nombreLu;
}

/// Permet de lire un character, après avoir afficher un message et
/// une liste de char en rapport avec la question, et de le renvoyer
/// \param msgPrompt une string d'affichage
/// \param affichageChars les valeurs prises en compte
```

```
77  /// \param afficherCharsVal boolean, true les char d'information sont affichés
78  /// \return le charactere saisi
79  char LireChar(const string& msgPrompt,
80                    const vector<char>& affichageChars,
81                    bool afficherCharsVal)
82  {
83      char charLu = ' ';
84
85      Afficher(msgPrompt, false);
86      Afficher(CARACTERE_ESPACE, false);
87
88     // Affiche les réponses possibles.
89      if (afficherCharsVal)
90      {
91          Afficher('[', false);
92          Afficher(affichageChars.front(), false);
93
94          for (vector<char>::const_iterator it = affichageChars.begin() + 1;
95                it != affichageChars.end();
96                ++it)
97          {
98              Afficher(" / ", false);
99              Afficher(*it, false);
100         }
101
102         Afficher("] ", false);
103     }
104
105     Afficher(CARACTERE_FIN_PROMPT, false);
106     Afficher(CARACTERE_ESPACE, false);
107
108     cin >> charLu;
109     cin.clear();
110     VIDER_BUFFER();
111
112     return charLu;
113 }
114
115 /// Affiche un message dans la console. Fonction surchargée
116 /// \param message Message à Afficher
117 /// \param retourLigne Est-ce qu'il faut faire un retour de ligne?
118 void Afficher(const string& message,
119                   bool retourLigne)
120 {
121     cout << message;
122
123     if (retourLigne)
124     {
125         cout << endl;
126     }
127 }
128
129 /// Affiche un caractère dans la console. Fonction surchargée
130 /// \param caractere Caractère à Afficher
131 /// \param retourLigne Est-ce qu'il faut faire un retour de ligne?
132 void Afficher(char caractere,
133                   bool retourLigne)
134 {
135     Afficher(string(1, caractere), retourLigne);
136 }
137
138
139 /// Surcharge de l'opérateur de flux afin d'afficher un vecteur.
140 /// \param os Flux de sortie
141 /// \param v Vecteur à afficher.
142 /// \return Référence au flux de sortie.
143 ostream& operator<<(ostream& os,
144                           const vector<int>& v)
145 {
146    os << '[';
147    for (vector<int>::const_iterator it = v.begin(); it != v.end(); ++it)
148     {
149       if (it != v.begin())
150          os << ", ";
151       os << *it;
152    }
```

```
153       return os << ']';
154   }
155
156   /// Permet de savoir si un char apparait dans une vecteur de char
157   /// \param c char a tester
158   /// \param listDeChar un vecteur de char
159   /// \return true si le char est dans la liste, false sinon
160   bool estCharDansVect(const char& c,
161                        const vector<char>& listDeChar)
162   {
163       return (*find(listDeChar.begin(), listDeChar.end(), c) == c);
164   }
```

```
/*
-----------------------------------------------------------------------------
Nom du fichier    : Dictionnaire.h
Nom du labo       : Labo08_Galton Groupe L
Auteur(s)         : Jeremiah Steiner, Ylli Fazlija
Date creation     : 11.01.2022
Description (But) : Fichier d'en-tête servant de recueil de constantes
                          - des caractères et des chaînes de caractères.
                    - Le principe de ce fichier est de pouvoir changer la langue
                    - ou les termes utilisés très facilement.
Remarque(s)       :
Compilateur       : Mingw-w64 g++ 8.1.0
-----------------------------------------------------------------------------
*/

#ifndef LABO08_GALTON_DICTIONNAIRE_H
#define LABO08_GALTON_DICTIONNAIRE_H

#include <string>
#include <vector>

const char CARACTERE_ESPACE             =   ' ';
const char CARACTERE_FIN_PROMPT         =   ':';
const char CARACTERE_AFFICHAGE          =   '*';
const char CARACTERE_ESPACEMENT         =   ' ';
const char CARACTERE_VALIDATION_1       =   'Y';
const char CARACTERE_VALIDATION_2       =   'y';
const char CARACTERE_REFUS_1             =  'N';

const std::string MESSAGE_PROMPT_HAUTEUR            = "Veuillez entrer un nombre "
                                                      "pour la hauteur";
const std::string MESSAGE_PROMPT_BILLES         = "Veuillez entrer un nombre "
                                                      "pour les billes";
const std::string MESSAGE_ERREUR_NOMBRE         = "Ce nombre n'est pas valable!";
const std::string MESSAGE_FIN                           = "Veuillez appuyer sur ENTER pour "
                                                      "quitter le programme...";
const std::string MESSAGE_DEMARRAGE                 = "Labo 08 - Galton";
const std::string MESSAGE_RECOMMENCER               = "Voulez-vous recommencer ?";

const std::vector<char> CARACTERES_VALIDATION_FIN = {CARACTERE_VALIDATION_1,
                                                     CARACTERE_VALIDATION_2};

const std::vector<char> CARACTERES_AFFICHAGE_FIN = {CARACTERE_VALIDATION_1,
                                                    CARACTERE_REFUS_1};

#endif //LABO08_GALTON_DICTIONNAIRE_H

```