```
/*
-----------------------------------------------------------------------------
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.
      tableauBilles[indice]++;
```

```
   }
}

/// Permets d'afficher une représentation graphique du tableau
/// à l'aide d'une double boucle.
/// \param CARACTERE Le caractère qui représentera une bille.
/// \param ESPACE Le caractère qui représentera les espaces.
void Galton::AfficherTableauGraphique(const char& CARACTERE,
                                      const char& ESPACE) const
{
   for (size_t i = ((size_t) *max_element(tableauBilles.begin(),
                                          tableauBilles.end())); i > 0; --i)
   {
      for (size_t val : tableauBilles)
      {
         cout << (val >= i ? CARACTERE : ESPACE);
      }
      cout << endl;
   }
}

/// Simple getter afin d'accéder au tableau de billes d'une instante Galton.
vector<int> Galton::getTableauBilles() const {
   return tableauBilles;
}
```