```
/*
 ------------------------------------------------------------------------------------
 Laboratoire : 04
 Fichier     : main.cpp
 Auteur(s)   : Thibaud Franchetti, Sacha Perdrizat
 Date        : 08.04.2019

 But         : Teste la classe générique Collection et la classe Produit

 Remarque(s) :

 Compilateur : GCC-g++ 7.3.0
               GCC-g++ 8.2.0
 ------------------------------------------------------------------------------------
*/

#include <cstdlib>
#include <list>
#include <vector>
#include "collection_g.h"
#include "exceptions.h"
#include "produit.h"

using namespace std;

int main() {

   {
      cout << "-----------------------------------------------------------"  << endl;
      cout << "Test sur Collection<char, vector> :" << endl;
      try {
         Collection<char, vector> c;
         for (char ch = 'A'; ch < 'D'; ++ch)
            c.ajouter(ch);
         cout << c << " (taille = " << c.taille() << ")" << endl;
         c.get(0) = 'B';
         c.get(1) = c.get(2);
         c.get(2) = 'D';
         cout << c << " (taille = " << c.taille() << ")" << endl;
         cout << boolalpha
              << c.contient('A') << endl
              << c.contient('D') << endl
              << noboolalpha;
         c.vider();
         cout << c << " (taille = " << c.taille() << ")" << endl;
         cout << c.get(0) << endl;
      } catch (const IndiceNonValide& e) {
         cout << e.what() << endl;
      }
      cout << "-----------------------------------------------------------"  << endl;
      cout << endl;
   }

   {
      cout << "-----------------------------------------------------------"  << endl;
      cout << "Test sur Produit :" << endl;
      try {
         // un produit se caractérise par un no, un libellé, un prix
         Produit p(1, "p", 0.05);
         cout << p << endl;
         {
            try {
               Produit p(1, "p", 0);
            } catch (const PrixNonValide& e) {
               cout << e.what() << endl;
            }
         }
         p.setPrix(0.0);
      } catch (const PrixNonValide& e) {
         cout << e.what() << endl;
      }
      cout << "-----------------------------------------------------------"  << endl;
      cout << endl;
   }

   {
      cout << "-----------------------------------------------------------"  << endl;
      cout << "Test sur Collection<Produit, list> :" << endl;
```

```
      try {
         Collection<Produit, list> c;
         Produit p1(1, "Produit 1", 1.55);
         Produit p2(2, "Produit 2", 5);
         c.ajouter(p1);
         c.ajouter(p2);
         cout << c << " (taille = " << c.taille() << ")" << endl;
         Produit tmp = c.get(0);
         c.get(0) = c.get(1);
         c.get(1) = tmp;
         cout << c << " (taille = " << c.taille() << ")" << endl;
         cout << boolalpha
              << c.contient(p1) << endl
              << c.contient(p2) << endl
              << noboolalpha;
         {
            auto majorationPrix = [](Produit& p){
               p.setPrix(p.getPrix() * 1.1);
               return p;
            };
            // On parcourt la collection en majorant le prix de chacun
            // des produits de 10%
            c.parcourir(majorationPrix);
            cout << c << " (taille = " << c.taille() << ")" << endl;
         }
         c.vider();
         cout << c << " (taille = " << c.taille() << ")" << endl;
      } catch (const IndiceNonValide& e) {
         cout << e.what() << endl;
      }
      cout << "-----------------------------------------------------------------"  << endl;
      cout << endl;
   }

   return EXIT_SUCCESS;
}

// -----------------------------------------------------------
// Test sur Collection<char, vector> :
// [A, B, C] (taille = 3)
// [B, C, D] (taille = 3)
// false
// true
// [] (taille = 0)
// Erreur dans Collection::get :
// n doit etre strictement plus petit que collection.size()
// -----------------------------------------------------------
//
// -----------------------------------------------------------
// Test sur Produit :
// (1, "p", 0.05)
// Erreur dans Produit::Produit :
// le prix doit etre >= 5 cts !
// Erreur dans Produit::setPrix :
// le prix doit etre >= 5 cts !
// -----------------------------------------------------------
//
// -----------------------------------------------------------
// Test sur Collection<Produit, list> :
// [(1, "Produit 1", 1.55), (2, "Produit 2", 5.00)] (taille = 2)
// [(2, "Produit 2", 5.00), (1, "Produit 1", 1.55)] (taille = 2)
// true
// true
// [(2, "Produit 2", 5.50), (1, "Produit 1", 1.71)] (taille = 2)
// [] (taille = 0)
// -----------------------------------------------------------
```