

```

/*
-----
Laboratoire : 04
Fichier      : collection_g_impl.h
Auteur(s)    : Thibaud Franchetti, Sacha Perdrizet
Date         : 08.04.2019

But          : Implémente la classe Collection déclarée dans collection_g.h

Remarque(s) :

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

#ifndef COLLECTION_G_IMPL_H
#define COLLECTION_G_IMPL_H

// Génère le message d'erreur à afficher à l'aide du paramètre nomFonction
template <typename T, template<typename, typename> class CONTENEUR>
std::string Collection<T, CONTENEUR>::indiceNonValide(const char* nomFonction){
    std::stringstream os;

    os << "Erreur dans Collection::" << nomFonction << " : " << std::endl
        << "n doit etre strictement plus petit que collection.size()";

    return os.str();
}

template <typename T, template<typename, typename> class CONTENEUR>
void Collection<T, CONTENEUR>::ajouter(const T& element) {
    data.push_back(element);
}

template <typename T, template<typename, typename> class CONTENEUR>
size_t Collection<T, CONTENEUR>::taille() const {
    return data.size();
}

template <typename T, template<typename, typename> class CONTENEUR>
T& Collection<T, CONTENEUR>::get(size_t pos) {
    if (pos >= taille()) {
        throw IndiceNonValide(indiceNonValide(__func__));
    }
    auto i = data.begin();
    for(size_t c = 0; c < pos; ++c){
        ++i;
    }
    return *i;
}

template <typename T, template<typename, typename> class CONTENEUR>
bool Collection<T, CONTENEUR>::contient(const T& element) const {
    return std::find(data.begin(), data.end(), element) != data.end();
}

template <typename T, template<typename, typename> class CONTENEUR>
void Collection<T, CONTENEUR>::vider() {
    data.clear();
}

template <typename T, template<typename, typename> class CONTENEUR>
template <typename UnaryOperator>
void Collection<T, CONTENEUR>::parcourir(const UnaryOperator& fonction) {
    std::transform(data.begin(), data.end(), data.begin(), fonction);
}

template <typename T, template<typename, typename> class CONTENEUR>
std::ostream& operator<< (std::ostream& os, const Collection<T, CONTENEUR>& c) {
    const T& dernier = c.data.back();

    os << '[';
    for (const T& i : c.data) {
        os << i << (i != &dernier ? ", " : "");
    }
    return os << ']';
}

```

```
#endif
```