```
/*
-----------------------------------------------------------------------------------

Nom du fichier    : vehicule.c
Auteur(s)         : Émilie Bressoud, Olin Bourquin, Timothée Van Hove
Date création     : 23.05.22
Description       : Implémentation des fonctions de vehicule.h

Remarque(s)       : utilisation de strncpy, ce qui permet de copier la chaine de caractères sans dépasser
                    la taille.
                    les enum sont casté en int pour pouvoir faire des comparaisons.
Compilateurs      : Apple clang 13.0.0 et MinGW-W64 11.2.0

-----------------------------------------------------------------------------------
*/

#include "vehicule.h"
#include <string.h> //strncpy()

Vehicule attributsVehicule(Vehicule* v, const char* immatriculation, const char* marque) {
    strncpy(v->immatriculation, immatriculation, TAILLE_MAX_IMMATRICULATION);
    strncpy(v->marque, marque, TAILLE_MAX_MARQUE);
    return *v;
}

Vehicule voitureHautDeGamme(const char* marque, const char* immatriculation, uint16_t puissance,
                            uint16_t poids) {

    VoitureHautdeGamme vhg = {puissance};
    Voiture v = {poids, HAUT_DE_GAMME, {.hautDeGamme = vhg}};
    Vehicule voiture = {"", "", VOITURE, {.voiture = v}};

    return attributsVehicule(&voiture, immatriculation, marque);
}

Vehicule voitureStandard(const char* marque, const char* immatriculation, uint16_t poids,
                         uint16_t cylindree, uint16_t co2) {

    VoitureStandard vs = {cylindree, co2};
    Voiture v = {poids, STANDARD, {.standard = vs}};
    Vehicule voiture = {"", "", VOITURE, {.voiture = v}};

    return attributsVehicule(&voiture, immatriculation, marque);
}

Vehicule camionnette(const char* marque, const char* immatriculation, double volume) {
    Camionnette c = {volume};
    Vehicule camionnette = {"", "", CAMIONNETTE, {.camionnette = c}};

    return attributsVehicule(&camionnette, immatriculation, marque);
}

int compTypeVehicules(const void* v1, const void* v2) {
    return (int) ((Vehicule*) v2)->typeVehicule - (int) ((Vehicule*) v1)->typeVehicule;
}

int compTypeVoitures(const void* v1, const void* v2) {
    return (int) ((Voiture*) v1)->typeVoiture - (int) ((Voiture*) v2)->typeVoiture;
}

size_t compterTypeVehicule(Vehicule* v, size_t nbVehicules, TypeVehicule type) {
    size_t nb = 0;
    for (size_t i = 0; i < nbVehicules; i++) {
        if (v[i].typeVehicule == type) {
            ++nb;
        }
    }
    return nb;
}

size_t compterTypeVoiture(Vehicule* v, size_t nbVehicules, TypeVoiture type) {
    size_t nb = 0;
```

```
    for (size_t i = 0; i < nbVehicules; i++) {
        if (v[i].typeVehicule == VOITURE && v[i].specificitesVehicule.voiture.typeVoiture == type) {
            ++nb;
        }
    }
    return nb;
}
```