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

Nom du fichier    : vehicule.h
Auteur(s)         : Émilie Bressoud, Olin Bourquin, Timothée Van Hove
Date création     : 23.05.22
Description       : fichier contenant la déclaration et la définition de la structure Vehicule,
                    des fonctions de création des 3 types de véhicules différents (voiture standard,
                    camionnette et voiture haut de gamme) donné par le struct Vehicule, des fonctions
                    permettant de compter le nombre de véhicules de chaque type et fonctions de
                    comparaisons entre types de véhicules.

Remarque(s)       : utilisation d'Union entre une voiture et une camionnette et entre une voiture de luxe
                    et une voiture standard. Cela permet d'économiser de la mémoire, sans pour autant
                    rendre le code moins lisible.
                    On suppose que les plaques d'immatriculation sont de la forme AA 000000 (Suisse).

Compilateurs      : Apple clang 13.0.0 et MinGW-W64 11.2.0

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

#ifndef TAXES_VEHICULE_H
#define TAXES_VEHICULE_H

#include <stdint.h> //uint16_t, size_t

#define TAILLE_MAX_MARQUE 20

// 2 caractères pour le canton, 1 caractère pour l'espace et 6 caractères maximum pour le numéro
// d'immatriculation en Suisse
#define TAILLE_MAX_IMMATRICULATION 9

typedef char Immatriculation[TAILLE_MAX_IMMATRICULATION + 1];
typedef char Marque[TAILLE_MAX_MARQUE + 1];

typedef enum {
    VOITURE, CAMIONNETTE
} TypeVehicule;

typedef enum {
    STANDARD, HAUT_DE_GAMME
} TypeVoiture;

typedef struct {
    uint16_t puissance; // en [CV]
} VoitureHautdeGamme;

typedef struct {
    uint16_t cylindree; // en [cm3]
    uint16_t co2; // en [g/km]
} VoitureStandard;

typedef union {
    VoitureStandard standard;
    VoitureHautdeGamme hautDeGamme;
} SpecificitesVoiture;

typedef struct {
    uint16_t poids; // en [kg]
    TypeVoiture typeVoiture;
    SpecificitesVoiture specificitesVoiture;
} Voiture;

typedef struct {
    double volume; // en [m3]
} Camionnette;

typedef union {
    Voiture voiture;
    Camionnette camionnette;
} SpecificitesVehicule;
```

```
typedef struct {
    Immatriculation immatriculation;
    Marque marque;
    TypeVehicule typeVehicule;
    SpecificitesVehicule specificitesVehicule;
} Vehicule;

//*************************************************************************************************
// Fonctions de création de véhicules
//*************************************************************************************************

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

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

Vehicule camionnette(const char* marque, const char* immatriculation, double volume);

//*************************************************************************************************
// Fonctions de comparaison et de comptage
//*************************************************************************************************

// Fonction permettant de construire l'immatriculation et la marque dans une structure Vehicule
Vehicule attributsVehicule(Vehicule* v, const char* immatriculation, const char* marque);

// Fonction de comparaison du type de vehicule entre 2 véhicules utilisés dans qsort
int compTypeVehicules(const void* v1, const void* v2);

// Fonction de comparaison du type de voitures entre 2 véhicules utilisés dans qsort
int compTypeVoitures(const void* v1, const void* v2);

// Compte le nombre d'éléments d'un sous-type de Vehicule dans un tableau
size_t compterTypeVehicule(Vehicule* v, size_t nbVehicules, TypeVehicule type);

// Compte le nombre d'éléments d'un sous type de Voiture dans un tableau
size_t compterTypeVoiture(Vehicule* v, size_t nbVehicules, TypeVoiture type);

#endif
```