SAE 101 : La gestion d'un parc de voitures
- Deuxième séance -

Objectifs :

- Implementation du paquetage *parking*
- Travailler le tableau d'enregistrements

Nous allons continuer l'application en Ada pour la gestion d'un parc de voitures en utilisant le paquetage *voiture* de la semaine dernière. Dans cette deuxième séance, nous aimerons programmer le paquetage *parking* qui comporte :

- La définition du type parking nommé *Type_Parking*
- Un ensemble de sous-programmes permettant la manipulation du *Type_Parking*.

Ainsi, vous allez construire les jeux de tests fonctionnels pour s'assurer que votre code satisfait le comportement attendu.

En ce qui concerne le dépôt de cette semaine : déposez les sources de votre projet Ada dans un dépôt Moodle à 18h30.

1 Modélisation du type *Type_Parking*

La structure de données *Type_Parking* permet de regrouper :

- un tableau nommé *voitures* qui permet de stocker *NB_MAX* voitures de type *Type_Voiture*, dont sa définition est :

```
LENGTH : constant Integer := 9;
type CharArray is array (1.. LENGTH) of Character;

type Type_Voiture is record
   num : CharArray ;
   place : Integer;
   heureEntree  : Integer;
end record;
```

- une valeur entière *nb_voitures* qui défini le nombre de voitures réelles stockées dans un parking à un instant donnée.

Proposer une structure de données complexe permettant de regrouper les plusieurs données du parking.

2 Programmer le paquetage *parking*

1. Créez les deux fichiers du paquetage *parking* (*parking.ads* et *parking.adb*).

Piqûre de rappel :

- La spécification du paquetage contient les definitions des types et les en-têtes des sous-programmes (fichier avec l'extension .ads)
- Le corps du paquetage contient les corps des sous-programmes (fichier avec l'extension .adb)

2. Définissez le type de donnée *Type_Parking*

   Nous allons maintenant à écrire un ensemble minimal de sous-programmes qui implémentent les opérations de manipulation du *Type_Parking*. N'oubliez de proposer (en parallèle avec l'implémentation des opérations) un programme principal *test_parking.adb* qui assure le comportement attendu des sous-programmes.

3. Écrire le sous-programme *remplir_parking* qui lit itérativement l'ensemble de voitures du parking *p* de type *Type_Parking*. Le nombre de voitures à lire n'est pas connu à priori. Le programme doit demander à l'utilisateur après chaque saisie s'il veut continuer à saisir des voitures. La lecture doit continuer tant que la saisie de la valeur soit "o"("O") et avant avoir saisi *NB_MAX* voitures.

4. Écrire le sous-programme *afficher_parking* qui affiche toutes les informations des voitures stationnées au parking *p* de type *Type_Parking*. La figure ci-dessous montre le résultat attendu à l'écran après l'execution du sous-programme *afficher_parking* qui contient deux voitures :

```
##############################################
############## Informations  #################
##############################################

La voiture avec numero d'immatriculation : 444555TTT
est stationee a la place de parking  :           2
et son heure d'arrivee est :           0h:         20m


##############################################
############## Informations  #################
##############################################

La voiture avec numero d'immatriculation : 444332AZT
est stationee a la place de parking  :         102
et son heure d'arrivee est :           0h:          0m
```

5. Modifiez le programme *test_parking.adb* afin de valider les sous-programmes *remplir_parking* et *afficher_parking*.

6. Écrire le sous-programme *trouver_voiture* qui recherche de façon séquentielle une voiture *v* dans un parking *p* de type *Type_Parking*. Le sous-programme renvoie VRAI si la voiture a été trouvée. Ainsi, le sous-programme renvoie la position *rang* de la voiture trouvée. Pour valider le sous-programme, vous devez effectuer la vérification dans différents scenarios dans le programme principal *test_parking.adb*.

7. Écrire le sous-programme *ajouter_voiture* qui insère une voiture *v* dans un parking *p* de type *Type_Parking*. La voiture est insérée après la dernière voiture ajoutée. Le sous-programme vérifie si la voiture *v* peut être insérée dans le parking. Tester

le sous-programme dans le programme principal *test_parking.adb*.

8. Écrire le sous-programme *ajouter_voiture_pos* qui insère une voiture *v* dans un parking *p* de type *Type_Parking*. La voiture est insérée à la position *pos* du tableau *voitures* du parking *p*. Le sous-programme vérifie si la voiture *v* peut être insérée dans la position *pos* du parking. Tester le sous-programme dans le programme principal *test_parking.adb*.

9. Écrire le sous-programme *supprimer_voiture* qui supprime la dernière voiture ajoutée au parking *p* de type *Type_Parking*. Le sous-programme vérifie si le parking contient des voitures. Tester le sous-programme dans le programme principal *test_parking.adb*.

10. Écrire le sous-programme *supprimer_voiture_pos* qui supprime la voiture stockée à la position *pos* du tableau *voitures* du parking *p* de type *Type_Parking* . Le sous-programme vérifie si la voiture stockée à la position *pos* du parking peut être supprimé. Tester le sous-programme dans le programme principal *test_parking.adb*.