



Système de Gestion des Parkings Lillois

Licence 1 – Semestre 2

Étudiants : BASTIN Léo & DUSSART Gabriel



ANNÉE UNIVERSITAIRE 2024 – 2025

 INSA <small>INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE</small>	LICENCE INFORMATIQUE	 Université Polytechnique <small>HAUTS-DE-FRANCE</small>
---	-----------------------------	---

Sommaire :

Table des matières

I. Introduction.....	3
II. Fonctionnalités Implémentées.....	4
1) Système d'entrée de parking.....	4
2) Vérification des places disponibles.....	5
III. Difficultés Rencontrées et Solutions Apportées.....	6
Difficultés.....	6
Solutions.....	6
IV. Commentaires et Suggestions.....	7
Remarque générales.....	7
Suggestions d'amélioration.....	7
Focus sur les Apprentissages.....	7
V. Auto Évaluation.....	8
VI. Code et Github.....	8

 INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE	LICENCE INFORMATIQUE	 Université Polytechnique HAUTS-DE-FRANCE
---	----------------------	--

I. Introduction

Le projet consiste à la gestion d'un systèmes de parkings, de pouvoir afficher un nombre indéfini de parkings, de noter l'entrée et la sortie d'un client d'un parking afin de calculer le prix à payer, de vérifier le nombre de places disponibles, ainsi que d'un mode administrateur pour pouvoir modifier un parking dans la liste.

Chaque question à sa fonction correspondante qui est appelée dans le main, qui s'agit d'un menu avec l'appel des différentes fonctions.

II. Fonctionnalités Implémentées

1) Système d'entrée de parking

```
void entrerParking(clients* tab_cl, int* num_clients, parking *tab)
{
    int i = 0;
    while (i < nb_personne)
    {
        if (num_clients[i] == 0)
        {
            printf("Entrer votre numero clients (different de 0) :\n");
            printf("Numero client :");
            scanf("%d", &num_clients[i]);
            printf("Entrer votre plaque d'immatriculation:\n");
            printf("Plaque d'immatriculation :");
            scanf("%s", tab_cl[i].plaque_immatriculation);
            printf("Entrer la date actuelle(de type JJ/MM/AAAA):\n");
            printf("Date actuelle :");
            scanf("%d/%d/%d", &tab_cl[i].date_enter.tm_mday, &tab_cl[i].date_enter.tm_mon, &tab_cl[i].date_enter.tm_year);
            printf("Entrer votre heure d'entrer(de type HH:MM):\n");
            printf("Heure entrer :");
            scanf("%d:%d", &tab_cl[i].heure_enterer.tm_hour, &tab_cl[i].heure_enterer.tm_min);
            break;
        }
        else
        {
            i++;
        }
    }
    mettreJourOccupation(tab);
    sauvegarderEtatParking(tab);
}
```

On a la fonction « entrerParking » qui s'occupe de l'entrée un client dans un parking spécifié. On a d'abord une boucle while, qui répète le code en boucle jusqu'à ce que nb_personne ait été atteint, puis une condition if, qui vérifie si la place dans le tableau num_clients est vide, sinon il cherche la place suivante.

Ensuite on demande à l'utilisateur d'entrer son numéro de client, sa plaque d'immatriculation, la date et l'heure actuelle., puis on enregistre chaque information dans une variable.

Pour finir, avec la fonction mettreJourOccupation, on met à jour la capacité d'un parking, décrétementée dans ce cas là pour enlever une place libre.



2) Vérification des places disponibles

```
int verifierPlacesDisponibles (parking *tab)
{
    int i = 1;
    char name[50];
    printf("Donnez l'identifiant du parking : ");
    scanf("%s", name);
    while (i < NB_PARKING && strcmp(tab[i].identifiant, name) != 0)
    {
        if ((atoi(tab[i].capacite_max) - atoi(tab[i].places_disponibles)) == 0)
        {
            return 1;
        }
        else
        {
            return 0;
        }
        i++;
    }
}
```

On a la fonction `verifierPlacesDisponibles` qui permet de comparer le nombre de places disponibles avec la capacité max d'un parking.

On demande à l'utilisateur de donner l'identifiant d'un parking, puis tant que tous les parkings n'ont pas été parcourus et que le parking spécifié n'a pas été trouvé, on convertit en entier la capacité max et le nombre de places disponibles, puis on les soustrait : si cette opération est égale à 0, on retourne 1 (pour dire qu'il est plein), sinon on retourne 0. Il n'y a pas besoin de spécifier le cas où le résultat est négatif, puisqu'il n'y aura jamais plus de places disponibles que de capacité max.

Cette fonction nous a permis de connaître la fonction `atoi` qui permet de convertir une chaîne de caractères en entier.

 INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE	LICENCE INFORMATIQUE	 Université Polytechnique HAUTS-DE-FRANCE
---	----------------------	--

III. Difficultés Rencontrées et Solutions Apportées

Difficultés

La principale difficulté rencontrée est en rapport avec la compilation du code, qui au début de chaque séance, ne fonctionnait pas à cause d'une erreur dans le code ou d'une erreur de notre part.

Une autre difficulté rencontrée est l'utilisation des différentes bibliothèques, comme la bibliothèque `time.h` pour la gestion des dates et `math.h` pour convertir une chaîne de caractères en hash.

À cause d'un problème dans le code, pas mal de temps a été consacré à la modification et l'enregistrement du fichier. Dans la fonction « `sortieParking` », la modification du fichier était mal faite, et donc n'enregistrait pas correctement les sorties.



Au cours de notre projet, le sujet a été changé sans avertissement au préalable, donc l'ordre des questions n'a pas été respecté et certaines questions ont été retirées.

Solutions

Comme solution pour la première difficulté, le chemin vers le code n'était pas forcément spécifié, ou que la commande de compilation n'était pas correcte, donc les solutions ne nécessitaient pas de modifier le code en lui même.

Nous avons aussi utilisé plusieurs forums comme StackOverflow ou Youtube afin de nous aider dans la résolution de bugs que nous avons rencontré.

Nous avons dû relire le code plusieurs fois et de réécrire certaines parties au vu de leur malfonctionnement, nous avons aussi, à une reprise, placés des `printf` à certains endroits afin de veiller au bon fonctionnement de la fonction `lesparkings` par rapport à l'ouverture du fichier.

 INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE	LICENCE INFORMATIQUE	 Université Polytechnique HAUTS-DE-FRANCE
---	----------------------	--

IV. Commentaires et Suggestions

Remarque générales

On peut remarquer que la question bonus n'a pas été faite car nous nous connaissions pas comment utiliser la bibliothèque.

Il y a aussi quelques bugs mineurs, comme l'affichage double de la plaque d'immatriculation dans le fichier clients.csv, ou l'utilisation obsolète de la bibliothèque pour convertir une chaîne en hash.



Suggestions d'amélioration

Par rapport à l'ancien sujet, le nouveau manque de certaines informations comme l'entrée du parking, qui est une fonction importante puisqu'elle permet de calculer le prix d'un utilisateur de manière plus simple et d'enregistrer pour le suivi des clients, et la création des structures qui facilite l'entièreté du projet.

Pour améliorer le projet, nous avons pu faire la question bonus, qui s'agit d'utiliser une bibliothèque pour faire une interface, mais par manque de connaissances et au vu de la complexité, nous n'avons pas pu le faire.

Focus sur les Apprentissages

Ce projet nous a appris plusieurs choses pour le langage C comme des bibliothèques (math.h, time.h) ou certaines fonctions comme sprintf qui retourne une chaîne formatée, scanf(" %[^\n]" ...) qui retourne une chaîne en enlevant les espaces, ou atoi (déjà spécifié dans les fonctionnalités implémentées). L'utilisation du langage C dans ce projet a permis d'en apprendre plus dessus, et donc d'améliorer ses compétences.

 INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES HAUTS-DE-FRANCE	LICENCE INFORMATIQUE	 Université Polytechnique HAUTS-DE-FRANCE
---	----------------------	--

V. Auto Évaluation

Sur l'ensemble du projet, nous avons trouvé que la difficulté variait : le sujet demandé n'était pas compliqué, mais la réalisation l'était.

En terme de planification, nous avons pu nous organiser : chacun travaillait de son côté, avant de mettre en commun lors des sessions de TP.

En terme de codage, cette partie était la plus compliquée, puisqu'elle demandait beaucoup de connaissances dans un langage de programmation dont nous étions pas forcément familier.

En terme de tests, chaque test a apporté des changements et des corrections de bugs.

Pour nous améliorer, nous devons apporter plus de connaissances dans le langage C, que ce soit en terme de connaissances de bibliothèques, de corrections de bugs, ou d'optimisation.

VI. Code et Github

Voici le lien pour le Github du projet :

https://github.com/LeoBastin380/langage_c/tree/main

Toutes les explications du code sont présentes dans le dépôt Github, il n'y a pas de dossier, puisqu'il n'y a pas de documentation hormis le readme.

Le dépôt se compose de 4 fichiers :

- le code principal Projet_parking.c
- le README.md
- le fichier clients.csv qui stocke les informations des clients
- le fichier parking-metropole.csv qui stocke les informations des parkings.