Anthony LE PORS

PEI L2 groupe TP2
Année 2020-2021

Projet en C – Application Covid

Sommaire

Introduction

Dans le cadre de l'unité d'enseignement GIF 1301, nous avons eu pour mission la création d'un programme en langage C. Le projet a débuté le 18 octobre 2020, avant les vacances de la Toussaint et avec une date limite de rendu le 7 décembre 2020, nous laissant 2 mois et demi pour réfléchir et travailler sur la réalisation du code. L'idée majeure de ce programme était de créer une « application anti-Covid », disposant de plusieurs fonctionnalités utiles à la gestion de la crise sanitaire.

Pour ce faire, nous avions 3 types, 3 structures imposées à utiliser dans notre projet. En effet, le rôle principal de ce projet est de vérifier notre maitrise des listes en langage C, qu'elles soient simplement ou doublement chaînées. Ces 3 structures nous permettront alors de créer la liste des citoyens, des lieux et des participants à chaque lieu.
L'utilisation de sous-programmes est-elle aussi nécessaire pour mener à bien ce projet.

C'est donc en alliant une performance des fonctions et un affichage donnant envie d'utiliser ce programme que l'application sera conçue. Il est important de bien cerner les contraintes de ce projet afin de commencer la conception de ce dernier. De plus, les méthodes de travail que j'ai utilisé seront décrites et expliquées dans ce rapport.
Des enseignements seront alors à conclure de la mise en œuvre de ce projet, auquel j'ai consacré beaucoup de temps durant ces 2 mois et demi.

Le développement qui suit retrace mes réflexions et mes analyses sur ce projet, énonçant ainsi les fonctionnalités présentes dans mon application.

Portée du projet

L'objectif du projet étant énoncé dans l'introduction, nous comprenons qu'une maitrise des listes en langage C est inévitable pour réussir ce projet.

L'enseignant nous a posé quelques contraintes afin d'encadrer le projet et d'évaluer notre niveau sur des points attendus. Outre les 3 types à utiliser, l'intérieur de ces structures est à analyser pour réfléchir à la conception du projet.

Dans un premier temps, en commençant par la structure *Tcitoyen*, je comprends que les citoyens seront définis par leur nom et prénom, avec pour chaque citoyen un pointeur qui pointe vers le citoyen précédent et un autre qui pointe vers le citoyen suivant.

La structure des lieux (*Tlieu*) est, elle, constituée d'une manière assez similaire : le nom du lieu qui est une chaine de caractères, un pointeur vers le début d'une liste de participants, un pointeur vers le lieu précédent et un autre vers le lieu suivant.

Enfin, la structure des participants (*Tparticipant*) est telle que nous y retrouvons dans chaque maillon de la liste : la date qui est stockée sous la forme d'un entier, un pointeur vers un élément de la structure *Tcitoyen* et un autre pointeur vers le participant d'après.

Dans un second temps, il y aussi plusieurs fonctionnalités de l'application qui sont attendues via un menu déposé à titre d'exemple par l'enseignant. En effet, il faudra que l'utilisateur puisse :

- Afficher la liste des citoyens
- Afficher la liste des lieux de fête
- Ajouter un citoyen
- Ajouter un lieu de fête
- Remplir les participants à une fête
- Afficher tous ceux qui ont rencontré un citoyen
- Enregistrer un fichier situation
- Ouvrir un fichier situation
- Supprimer tous les évènements antérieurs à une date

Une fois que toutes ces fonctionnalités seront atteintes en utilisant les contraintes, alors les attentes de ce projet seront remplies, tout en laissant cependant une certaine liberté pour améliorer l'application.

Analyse du projet

Une fois l'objectif du projet énoncé, avec la connaissance des contraintes, de nombreuses simplifications apparaissent, ce qui me donne déjà des pistes de départ.

En effet, il faut commencer par gérer les données concernant les lieux et les citoyens. Comme ces 2 structures sont doublement chainées, alors ce qui est fait pour l'une est aussi fonctionnelle pour l'autre, en adaptant seulement l'écriture pour bien utiliser les variables déclarées dans les types correspondants. Je remarque ici une symétrie importante dans le projet, ce qui me permettra de gagner du temps lors de la conception. Je peux alors partir d'exercices traités en cours en les adaptant ici au projet, ce qui me donne des idées quant à la manipulation des listes et les fonctions indispensables qui y sont associées.

Il restera ensuite une grosse partie liée à la gestion des participants, qui nécessitera pas mal de réflexion pour aboutir à quelque chose de performant. La première difficulté étant de comprendre comment sélectionner un lieu pour ensuite y ajouter un citoyen dans la liste de participant, le tout en choisissant sa date de de participation.

Une fois cette étape finie, il faut s'attaquer à la récupération des cas contacts d'un certain citoyen s'il est déclaré comme malade. Cette partie implique une forte maitrise de la gestion des dates. Est-il possible de rentrer une date sous forme de chaine de caractères avec des « / », puis convertir cette chaine en un entier ? Quel format faudra-t-il utiliser pour faciliter la soustraction de 7 jours à cette date afin de trouver les personnes vues dans la période de contamination ? Je me suis posé de nombreuses questions au moment où j'ai analysé cette partie, en étant parfois même pas sure d'aboutir à quelque chose de performant.
Cependant, j'aime bien gérer les différentes erreurs que peut commettre l'utilisateur, en essayant de réduire au maximum la possibilité de faire planter le programme. La réflexion de cette partie a donc été intense, et les choix retenus seront donc expliqués dans la conception du programme.

Enfin, en parallèle de tout cela, il est nécessaire d'enregistrer les données saisies dans l'application, pour qu'elle puisse être réutilisable. La création d'un fichier Situation implique la maîtrise de 2 parties. La première concerne l'écriture de données dans un fichier pour pouvoir le sauvegarder. La deuxième permet le chemin inverse, à savoir lire un fichier pour charger des données dans l'application. Une fois cette partie fonctionnelle, le programme sera bien plus facile d'utilisation, disposant de sa propre « base de données ».

Conception du programme

La compréhension de ce projet étant totale, la conception a pu se faire en plusieurs parties. Je vais ici essayer d'expliquer synthétiquement dans quel ordre j'ai conçu mon programme, en expliquant les avancées de ce dernier dans le temps.

Tout commence pendant les vacances de la Toussaint, ayant déjà fait des exercices en TP sur les listes doublement chaînées, je me suis occupé d'adapter ces exercices pour créer mes listes de citoyens et de lieux.

Durant tout le projet, une liste sera créée par ses bidons de début et de fin de liste. Ceux-ci ne contiendront aucune information utile mais ils permettront de ne pas sortir de la liste si on veut se balader dans celle-ci. De plus, le bidon de début pourra être utilisé pour définir la liste.

À ce moment-là, la console était toute noire, avec du texte blanc nous proposant 4 choix au menu :

- Afficher la liste des citoyens
- Afficher la liste des lieux
- Entrer dans le MENU Citoyen (création, ajout, suppression)
- Entrer dans le MENU Lieu (création, ajout, suppression)

La création de liste se faisait avec cette interface :

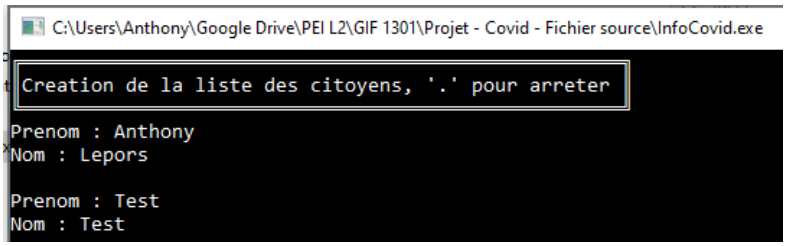

Figure 1 : Interface basique de création de liste

Pour supprimer un citoyen/lieu de cette liste, qui est une fonctionnalité que j'ai ajouté par rapport à celles attendues, il fallait réécrire le prénom et le nom à l'identique, ce qui est peu pratique.

Au moment où je commençais à gérer le remplissage des participants à un lieu, je me suis dit qu'il fallait que je consacre plus de temps que cela à la gestion de l'interface Homme/Machine. Mon idée première était de m'occuper de l'affichage de l'application qu'une fois que toutes les fonctionnalités attendues étaient opérationnelles. Cependant, je me suis rendu compte qu'il ne fallait pas procéder ainsi.

En effet, j'aurai dû reprendre tout le projet depuis le départ, puisque l'affichage d'une fonction doit être conçu en même temps que le code en lui-même de la fonction. Pendant le début du mois de novembre, je me suis consacré à la gestion de l'interface Homme/Machine, mettant alors le développement des fonctions du projet en pause.

Pour cela, j'ai commencé par maitriser la console Windows grâce à des fonctions trouvées sur le site même de Microsoft : https://docs.microsoft.com/fr-fr/windows/console.

Les premiers éléments à maitriser sur le développement de l'interface Homme/Machine ont alors été les suivants :

- Gestion d'affichage à l'endroit souhaité sur la console
- Gestion de la couleur du texte (couleur de fond + couleur caractère)
- Gestion de la console :
    - Titres de console
    - Taille de console
    - Fixation de la taille de console, à savoir empêcher l'utilisateur de modifier la taille de celle-ci

Je n'y avais pas pensé mais, une fois ces premiers éléments maitrisés, j'ai pris encore plus de plaisir à développer mon programme, son utilisation étant plus fluide. Je suis même revenu sur l'interface de la création de mes listes Citoyens et Lieux, en décidant de tout faire sur le même écran. L'interface était alors beaucoup plus agréable puisque la création, l'insertion et la suppression d'un citoyen/lieu dans la liste se faisait dans 1 seul menu.

J'en étais alors rendu à concevoir le remplissage de participants à des lieux. Pour cela, j'ai commencé par imaginer mon affichage en prenant une feuille blanche et en dessinant mon menu.

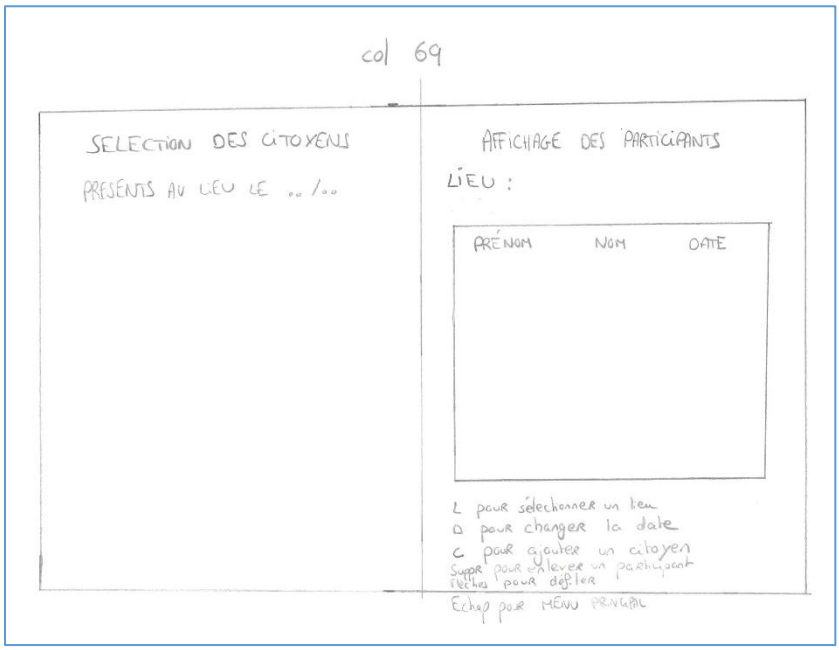

Figure 2 : Concept de l'interface pour le remplissage des participants

Je me suis dit qu'un des moyens les plus pratiques pour cela était de divisé l'écran en 2. La partie droite de l'écran sera toujours occupée par l'affichage des participants au lieu sélectionné. Mais les choix au clavier permettront de sélectionner un citoyen, une date et un lieu afin de créer la liste des participants. Cela apparaitra sur l'écran de gauche. J'ai ensuite essayé de rendre cela pratique d'utilisation pour ne pas déstabiliser l'utilisateur avec cet affichage qui change des précédents, avec notamment une bordure qui entoure la partie gauche ou droite suivant le choix de l'utilisateur.

Au cours de la réalisation de cette fonctionnalité majeure de l'application, je me suis occupé de l'enregistrement des données dans un fichier de sauvegarde. En effet, comme tout ne fonctionne pas

du premier coup, il est nécessaire de faire des tests. Et pour cela, c'est un gain de temps si les données sont déjà rentrées lorsqu'on réexécute l'application. Je me suis alors mis à comprendre comment enregistrer du texte dans un fichier. J'ai fait le choix de mettre du texte dans le fichier et non du binaire afin que ce fichier reste lisible et modifiable avec un éditeur de texte.

J'ai commencé par la sauvegarde des données, qui est la partie la plus simple, en écrivant dans un fichier. J'écris dans le fichier de la manière suivante :

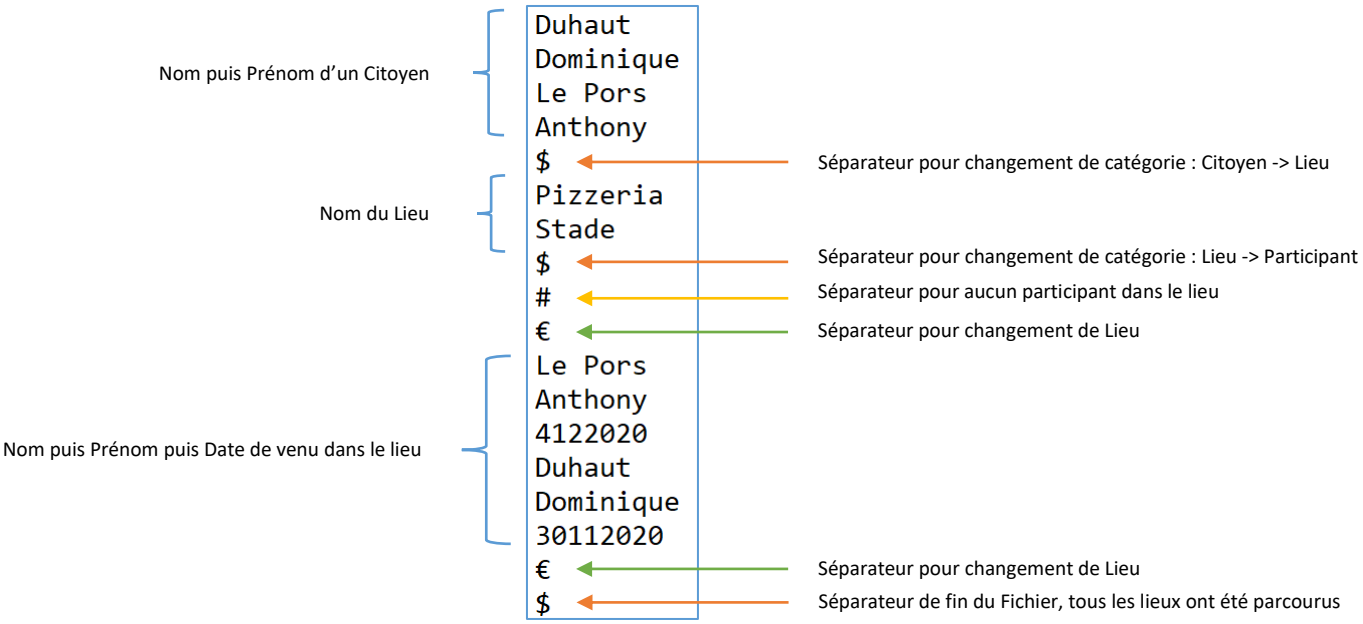

Figure 3 : Mise en forme du Fichier Situation

Ensuite, j'ai appris à charger le fichier en lisant le texte qu'il contenait. Il suffit de le faire ligne par ligne, en testant ce qui a été lu dans le fichier.

Enfin, pendant les 2 dernières semaines de novembre, le projet était déjà quasiment fonctionnel, il ne me manquait plus que la programmation de la partie finale, à savoir retrouver la liste des cas contacts. Je me suis inspiré de mon affichage en Figure 2 en divisant encore l'écran de la console, avec à droite la liste des cas contacts et à gauche, le choix d'un citoyen et l'insertion de sa date de contamination.

Pour rechercher les cas contacts, j'ai créé une structure Tdate qui me permet de rentrer toutes les dates à laquelle le citoyen malade est venu dans un lieu. Elle est utile dans le cas où le citoyen malade est venu plusieurs fois dans un même lieu. De plus, il m'a fallu créer de nombreuses fonctions pour maitriser au mieux les dates et ses formats. J'ai choisi d'utiliser le format français, à savoir « JJ/MM/AAAA ». C'est donc une chaîne de caractère qui est rentrée par l'utilisateur, je la convertis ensuite en entier du type JJMMAAAA. J'ai aussi décidé de créer un format assez spécial, qui est du type « AAAAnum_jour ». Ce format est bien plus pratique pour pouvoir soustraire 7 jours ou additionner 15 jours à cette date. En effet, num_jour correspond au numéro du jour dans l'année, prenant en compte la possibilité que l'année soit bissextile.

De plus, lorsque l'utilisateur rentre une date au clavier, je m'assure qu'elle est bien du type « JJ/MM/AAAA » grâce à une fonction de vérification. Si la date n'est pas valide, il faut retaper une nouvelle date.

Dans la liste des cas contacts, il apparait tous citoyens que le malade a vu les 7 derniers jours avant sa date de contamination, mais aussi les 15 jours suivants sa date de contamination. J'ai donc supposé que la maladie puisse être présente de manière asymptomatique pendant 7 jours, avec un malade qui reste contagieux jusqu'à 15 jours après la détection du virus.

Durant toute la conception, j'ai souvent trouvé des petits détails à améliorer et notamment au niveau du visuel pour améliorer l'interface Homme/Machine. La conception était donc très complète puisque je m'occupais à la fois de la programmation des fonctions et de la programmation de l'affichage. Une fois le programme terminé, je suis revenu sur tout mon travail pour ajouter des commentaires manquants et pour essayer d'optimiser certains trucs, dans le but de raccourcir le code, de faciliter sa relecture mais aussi d'améliorer la performance du programme lors de son exécution.

Méthodes de test

Pendant la conception du projet, je n'ai pas toujours travaillé dans le même fichier source. En effet, ce n'était pas idéal à cause de la longueur de mon code (≈3000 lignes). Je me suis donc créé un dossier « Test » en parallèle qui contient plusieurs fichiers sources. J'ai donc réalisé des tests unitaires pour divers éléments quand je savais que ce n'allait pas être facile.

Les différents fichiers source que je créais dans le répertoire Test était plus propice aux essais. En effet, je voyais de cette manière mieux ce que je faisais et j'utilisais parfaitement la notion de portabilité d'un sous-programme. En effet, je copiais les sous-programmes de mon programme principal vers le fichier de test, en sélectionnant que ceux qui m'étaient utiles. C'est pourquoi il est très important d'écrire un sous-programme de manière à ce qu'il soit portable et que toutes les variables soient définies dedans, soit en argument, soit en variable locale.

Il m'arrivait donc de créer des nouvelles fonctions dans mes fichiers tests, qu'il fallait ensuite copier dans mon programme principal une fois que je m'étais assuré de leur fonctionnement. A ce moment-là, il était donc primordial de faire un test d'intégration de ma nouvelle fonction. Je n'ai pas hésité à passer du temps pour vérifier et tester ma fonction dans les 2 programmes. En effet, il est aussi important de vérifier que le sous-programme est bien fonctionnel dans le programme complet. Il arrive parfois qu'un élément rentre en collision avec un autre et ce problème n'apparait qu'une fois dans le programme principal, alors qu'on croyait que notre sous-programme était parfaitement fonctionnel en le testant dans un autre répertoire. C'est pourquoi le test d'intégration est aussi important que le test en lui-même d'une fonction.

À la fin de ce projet, mon dossier Test contient donc plusieurs fichiers sources, les éléments principaux où j'ai utilisé cette méthode sont les suivants :

- Test des fonctions agissant sur la console
- Test de lecture des caractères
- Test des couleurs de console
- Test tri de liste
- Test de formats de date
- Test organisation du menu remplissage des participants

En fin de compte, il y a donc plusieurs points qui ont été testés dans un autre fichier avant de l'inclure au programme complet. C'est une rigueur de travail à ne pas négliger car elle aboutit souvent à un gain de temps.

En complément, j'ai aussi eu le besoin de créer un fichier Excel afin de tester mes fonctions de création de liste de cas contacts. J'y ai inséré dedans mes listes de participants à chaque lieu et je faisais des simulations manuelles. Cela me permettait ensuite de comparer ces 2 listes pour voir si mon programme était complet ou non. C'est grâce à cette méthode que je me suis rendu compte que si un citoyen venait plusieurs fois au même lieu, mon programme ne ressortait pas tous les participants rencontrés dans un premier temps.

J'ai aussi fait testé mon programme à plusieurs personnes qui ne connaissent pas la programmation, afin d'avoir un avis critique et pour avoir des idées ce que je pourrai améliorer, notamment sur l'interface Homme/Machine.

Résultats finaux du projet

Enfin, une fois les étapes d'analyse et de conception passées, nous pouvons maintenant tester le programme.

Tout d'abord, une fois exécuté, le programme propose à l'utilisateur de charger ou non une simulation. Je conseille à l'utilisateur d'ouvrir une première fois la simulation déjà existante, puis d'ensuite créer la sienne une fois qu'il s'est familiarisé à l'environnement. Il peut aussi aller dans le menu « Conseils d'utilisation » s'il veut un guide.

Lorsqu'une simulation est créée, il est possible de gérer les citoyens et les lieux, avec la faculté d'ajouter, d'éditer ou de supprimer un citoyen/lieu. Si un citoyen ou un lieu est supprimé par erreur, il est possible de le restaurer ou de le supprimer définitivement en entrant dans la poubelle correspondante.

Ensuite, l'utilisateur peut commencer à remplir ses listes de participants à différents lieux, en insérant des citoyens à une date choisie. Ce menu est très pratique puisqu'il est assez puissant pour gérer les participants à chaque lieu.

Une fois les participants insérés, il est donc possible d'afficher les cas contacts à n'importe quel citoyen. Il sera alors affiché tous les citoyens qu'il a vu 7 jours avant sa date de contamination mais aussi 15 jours après.

Il est aussi possible de supprimer tous les événements antérieurs à une date, cela supprimera alors tous les participants dans chaque lieu qui sont venus avant une date rentrée au clavier.

Il n'y a pas vraiment d'ordre à respecter, le programme est fait en sorte qu'il ne plante pas, ou du moins il est difficile de le faire bugger. Une utilisation normale de ce programme devrait alors se faire sans soucis par n'importe quel utilisateur.

Quand l'utilisateur quitte le programme, il a le choix de sauvegarder sa simulation, ou de quitter sans sauvegarder. Il est aussi possible d'enregistrer son fichier au cours de l'utilisation de l'application, via le menu de gestion du fichier Situation.

Perspectives (défauts et améliorations)

Bien que ce programme soit fini et assez complet, il lui reste encore quelques défauts. La prise en main de ce programme peut être assez compliquée pour une personne habituée à utiliser l'informatique de manière classique et quotidienne. L'utilisateur de ce programme peut être assez désorienté de ne devoir utiliser que le clavier. En effet, ne pas utiliser la souris peut être un peu frustrant, bien que les raccourcis au clavier soient souvent plus rapides.

De plus, lorsqu'une saisie au clavier est lancée par la fonction fgets, alors il n'est plus possible de revenir en arrière, ce qui est parfois embêtant quand l'utilisateur s'est trompé de fonctionnalité. Il existe cependant des alternatives à cela avec la possibilité d'écrire n'importe quoi et de supprimer après.

Aussi, il serait bien que la suppression d'un citoyen de la structure Tcitoyen supprime aussi ce citoyen de la liste de participants des lieux auquel il est venu, ce qui n'est actuellement pas le cas. Le citoyen reste en effet visible dans les participants.

En complément de ces défauts, ce sont surtout des idées qu'il me reste et que je n'ai pas eu le temps d'exploiter.

Il me manque la réalisation de fonctions permettant de trier une liste par ordre alphabétique à la lecture du fichier situation. En effet, la base de données peut se construire à partir du fichier situation plutôt que depuis l'application. Si c'est le cas, alors la liste ne sera pas triée par ordre alphabétique puisque je n'ai pas fait de fonctions dans cette éventualité qui n'est évidemment pas le cas courant. Le tri par ordre alphabétique fonctionne très bien du moment que l'utilisateur insère les données à partir de l'application.

De même, il serait surement utile au moment du remplissage de participants à des lieux de pouvoir ajouter un nouveau lieu ou un nouveau citoyen. Ce n'est pas compliqué à mettre en place et cela éviterait un retour au menu principal si l'utilisateur se rend compte qu'il lui manque un lieu ou un citoyen à ajouter.

Une gestion de poubelle de la même manière que pour les citoyens et les lieux est envisageable pour les participants, avec ici aussi la possibilité de restauration qui serait un plus pour l'utilisateur.

Enfin, ma plus grosse idée était de pouvoir sélectionner plusieurs citoyens malades en même temps, et de voir ainsi tous les cas contacts du Covid 19, ce qui m'aurait permis de faire des statistiques sur le pourcentage de la population potentiellement touchée par le virus. Celle-ci nécessiterait sans doute un temps conséquent, c'est pourquoi elle est restée au stade d'idée.

Conclusion

Ce projet s'est révélé très enrichissant dans la mesure où nous devions travailler sur un cas parfaitement dans le thème actuel.

Je pense avoir acquis de nombreuses compétences puisque je me sens maintenant très à l'aise vis-à-vis de l'utilisation des listes chaînées en C. La manipulation de pointeurs qui n'était pas forcément évidente pour moi au début est maintenant maîtrisée.

Je suis plutôt satisfait du résultat de ce projet, ce qui ne me fait pas regretter mon investissement au cours de ces 2 mois et demi de travail.

Toutefois, ce projet ne reste quand même qu'un projet en langage C, il ne présente pas vraiment d'avantage par rapport à une utilisation de base de données en langage SQL comme nous avions déjà pu le faire auparavant. Il ne faut donc pas penser avoir développé une réelle application « Anti-Covid ».

Annexe

J'ai créé un dossier qui contient 7 fichiers : le rapport, la source du code commenté à ouvrir avec Dev-C++, l'exécutable de l'application, l'icone de l'application et 3 fichiers de sauvegarde des données (contenu de l'application à savoir citoyens, lieux et participants ainsi que 2 fichiers de récupération de la poubelle des citoyens et des lieux supprimés).

Je vous souhaite donc une bonne utilisation de mon application Covid.