Projet C

Covid-19

Par TekneX

Sommaire :

Introduction

Le projet de l'UE C de l'année 2020 était en rapport avec l'actualité : il fallait concevoir un programme permettant de retrouver les cas contacts d'un malade. Pour ce faire nous disposions de trois chaînes :

- une chaîne permettant de lister tout les citoyens

- une autre pour lister les lieux d'une ville

- une dernière pour lier les citoyens a une fête dans un certain lieu, à une certaine date.

Nous avions à réaliser 9 tâches, allant de l'affichage des citoyens, à la suppression d'événements à partir d'une certaine date, en passant par l'enregistrement de fichiers situations ou encore l'affichage des personnes ayant rencontrés un certain citoyen. C'est d'ailleurs cette dernière tâche qui était la principale, puisqu'elle permettait de retrouver les cas contacts d'un citoyen.

J'ai donc crée un programme faisant fonctionner toutes ces tâches, et ce dossier en explique le fonctionnement.

Portée du projet

L'objectif de ce programme était de retrouver les cas contacts d'un malade. Nous disposions pour cela de listes (en utilisant des pointeurs), de sous-programmes, de fonctions à créer et des cours donné par notre Professeur. Voici la listes de fonction à réaliser pour terminer le projet :

- Afficher la liste des citoyens
- Afficher la liste des lieux de fêtes
- Ajouter un citoyen manuellement
- Ajouter un lieu de fêtes manuellement
- Remplir les participants a une fête
- Afficher tous ceux qui ont rencontre un citoyen
- Enregistrer un fichier situation pour une ville
- Ouvrir un ficher situation pour récupérer les citoyens/dates/participants
- Supprimer tous les événements antérieurs a une date, donnée par l'utilisateur

Notre Professeur nous avait également donné trois structures à respecter. J'ai choisi de mettre ces struct dans un dossier et de lier mon programme et ces struct avec un appel de ce fichier (#include "Fonctions/typeCovid2020.c").

Une fonction avait pour but de créer plusieurs fichiers Situations en fonction des villes pour pouvoir les récupérer par la suite sans avoir à ré-entrer chaque citoyens, chaque lieux et chaque éventements. J'ai décidé pour ce faire d'utiliser des signes distinctifs, comme £ et $ pour pouvoir arranger mes valeurs dans ces fichiers situations.

Malheureusement j'ai dut implémenter une contrainte majeure. Lors des fonctions pour créer et ouvrir les fichiers situations, j'ai choisis d'utiliser des dossiers. Le langage C permettant de créer des fichiers mais pas des dossiers, j'ai malheureusement dut utiliser des fonctions de terminal Windows, comme md ou rd. C'est donc a mon grand regret que mon programme n'est exécutable que sur Windows, et non sur Linux ou MacOS.

Une deuxième contrainte est maintenant liée a l'affichage. Pour que l'interface homme-machine soit plus esthétique, j'ai choisis de centrer certaines lignes au milieu de la console. Pour ce faire, j'ai choisis d'utiliser la taille de ma console. Malheureusement, si l'utilisateur met en plein écran la console ou que sa console personnelle à des proportions différentes, le centrage ne sera pas bien effectué, et l'interface en sera moins belle.

Mon programme a donc des conditions nécessaires pour être exécuté : être sur Windows et avoir une console de 120 caractères sur 30. Ces conditions sont fâcheuses, mais je n'ai pas réussi à passer au travers.

Analyse du projet

Il existe, dans mon programme, de nombreuse optimisations à effectuer et beaucoup de simplifications possibles. Malheureusement, ma faible connaissance du langage C ne pas m'a pas vraiment permis de réaliser ces améliorations. Mon programme contient en effet de nombreuses lignes, ce qui cause donc une plus grande place dans la mémoire et un plus long temps de compilation. Je pense aussi que ma mémoire n'est pas parfaitement bien géré : je pense principalement aux bump de maillons lors de la suppression d'événements. Malheureusement je n'ai pas les connaissances requises pour effectuer ces opérations.

Autrement, il existe de nombreuses améliorations possibles autre que dans le code pur. On pourrait par exemple améliorer l'affichage, qui est très basique, utiliser des fenêtres qui s'ouvrent, des onglets pour pouvoir plus facilement accéder au listes de citoyens et de lieux, modifier les couleurs…

De nombreuses améliorations sont possibles, et si j'avais eu assez de temps (car ces améliorations demandent énormément de temps) j'aurais put les mettre en place.

L'application TousAntiCovid créé par le Gouvernement est une des autres solutions permettant de connaître les cas contacts d'une victime du Covid-19, elle est plus efficace et plus jolie que mon programme. Si je voulais améliorer l'affichage de mon programme, je pourrais m'en inspirer.

Conception

Découpage du projet :

Ce projet a été découpé en plusieurs phases. J'ai d'abord créé toutes les fonctions, puis j'ai amélioré l'affichage à la fin, lorsque mon projet était entièrement terminé. J'ai utilisé pour chaque fonction des sous-programmes, pour améliorer la clarté du programme.

Utilisation :

Lorsque vous exécutez le programme, un menu va s'afficher.

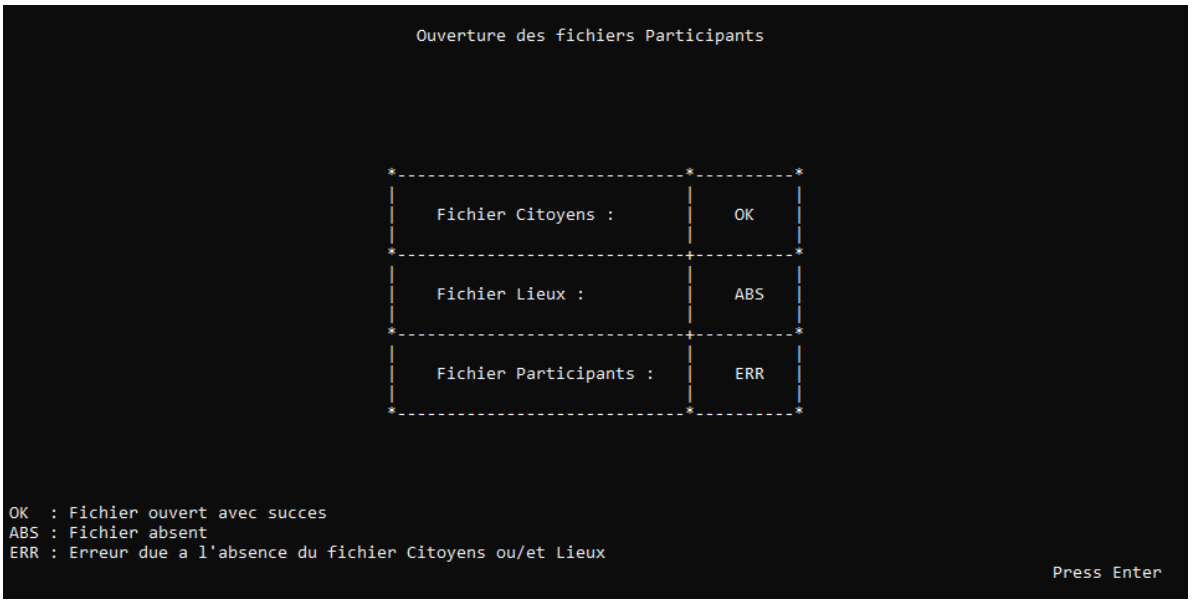
Vous pourrez ensuite choisir la fonction que vous voulez lancez. Pour cela, utilisez le pavé numérique. A la fin d'une fonction, appuyez sur Entrée ou sur Echap pour fermer la fonction et revenir au menu. Si vous voulez ensuite quitter le programme, appuyez sur la touche Echap.

Vous aurez plusieurs fois l'occasion d'utiliser un menu déroulant. Pour voyager à l'intérieur, utilisez les flèches directionnelles, puis appuyez sur Entrée pour confirmer votre choix.

Vous pourrez également créer des fichiers situations et les ouvrir. Pour la création il vous sera demandé de choisir une ville (symbolisée par un profil). Choisissez-en un déjà existant pour l'écraser ou entrez un nouveaux nom de profil pour en créer un.

Pour l’ouverture de fichiers Situations, ils vous sera également demandé d’entrée le nom d’un profil déjà existant. Veillez à ne pas faire de fautes de frappes ou le programme ne trouvera pas les fichiers. Vous pourrez retrouver ces fichiers dans le dossier /Fichiers_Situations.

Exemple d’un cas où le fichier lieux est manquant :



L’affichage utilise des fonctions nécessitant une console de 120*30 caractères. Veillez donc à ne pas mettre en plein écran la console, pour profiter du meilleur affichage.

Codage en C

Mon programme contient plusieurs types de fonctions :

- les fonctions d'interfaces : (exemples annexe p13)

    centre() : sert à centrer une chaîne de caractères

    bas_droite() : sert à mettre un chaîne de caractères en bas à droite

    gotoligcol() : sert a placer le curseur à un certain endroit de la console

    afficheMenu() : sert à afficher le menu principale

- les fonctions d'affichage de listes : (exemples annexe p13)

    AfficherCitoyens() : Affiche la liste des citoyens

    AfficherLieux() : Affiche la liste des lieux

    AfficherParticipants() : Affiche la liste des participants

    Rencontre() : Affiche la liste des citoyens ayant rencontrée quelqu'un

- les fonctions d'ajouts : (exemples annexe p14)

    AjoutCitoyen() : Ajout d'un citoyen

    AjoutLieux() : Ajout d'un lieux

    RemplirParticipants() : Ajout d'un citoyen à une fête, à une certaine date

- les fonctions d'enregistrement et d'ouverture de fichiers : (exemples annexe p14)

    EnregistreSituation() : Enregistre les fichiers situations

    OuvrirSituation() : Ouvre les fichiers situations

- les fonctions de suppressions : (exemples annexe p15)

    SupprimerApresDate() : Supprime les maillons d'une chaîne après une date

Il existe aussi 2 fonctions : WaitPress() qui permet de stopper le programme tant que la touche Entrée ou Echap n'est pas pressé, et Appui, qui renvoie la valeur ASCII d'une touche pressée. (exemples annexe p15)

Phases de tests

Pour tester les différentes parties de mon programme, j'ai utilisé différents moyens pour vérifier mes erreurs :

- Tout d'abord le printf(). En mettant des printf() avec 1,2,3… ou avec certaines valeurs comme pcitoyens→nom, on peux voir très concrètement où se trouve une erreur, une boucle infinie…

- Ensuite j'ai également créé une fonction permettant de visualiser les relations entre lieux, participants et citoyens. J'ai ensuite choisis d'implémenter cette fonction dans mon menu, en améliorant l'affichage pour que l'utilisateur puisse voir ces relations.

- J'ai aussi plusieurs fois utilisé la fonction WaitPress() que j'avais créé, pour avancer pas à pas dans une boucle par exemple.

- Enfin, j'ai aussi utilisé la fonction Appui() qui renvoie la valeur ASCII d'une touche appuyé. Pour cela il fallait appuyer sur p et ensuite presser la touche. J'ai commenté la partie d'appui de p par la suite, pour éviter à l'utilisateur de ne pas comprendre pourquoi des valeurs inconnues s'affichent.

Résultats

Pour terminer, toutes les fonctions du programme fonctionnent et aucun bug n'est présent. La fonction principale pour afficher les cas contacts d'un malade fonctionne et donne une liste des citoyens, en fonction du lieux et de la date :

```
                                Liste des gens ayant rencontre une personne


      Citoyen : Turmel Florian

Au lieu Lorient le 101020 :

- Folligne Gwenaelle
- Lataste Remy

Au lieu Rennes le 201021 :

- Grimbert Florian

                                                                        Press Enter
```

La fonction de suppression marche aussi, on obtient :

```
                                    Suppression des evenements



  Choix de la date a partir de laquelle supprimer :
     Jour :  20
     Mois :  10
     Annee : 20
Tous les evenements anterieurs au 20/10/20 ont ete supprimes
```

Enfin, la sauvegarde de fichiers situations nous donne ces trois fichiers :

Citoyens.dat

```
Turmel
Florian
Turmel
Isabelle
Turmel
Natacha
Folligne
Gwenaelle
Grimbert
Florian
Lataste
Remy
Macron
Manu
$
```

Lieux.dat

```
Lorient
Lille
Rouen
Paris
Caen
Rennes
$
```

*Participants.dat - Bloc-notes

```
101020
Turmel
Florian
101020
Folligne
Gwenaelle
101020
Lataste
Remy
£
91020
Grimbert
Florian
£
£
91020
Macron
Manu
```

Conclusion

J’ai trouvé ce projet très intéressant, d’un part car j’aime beaucoup la programmation et d’une autre part car je ne m’y connaissait pas du tout en C, et ça m’a permis d’apprendre les bases d’un nouveau langage. J’ai mis beaucoup de temps dans ce projet, d’abord car il m’était important qu’il soit bien réalisé, et d’une autre part car j’ai eu beaucoup de bugs que je n’arrivais pas à résoudre. Il m’est arrivé de rester bloquer sur une simple ligne qui n’arrivait pas à s’exécuter, parfois pendant plusieurs heures, mais je trouvais tout le temps la solution.

Pour conclure j’ai beaucoup aimé ce projet, le sujet était intéressant et j’ai obtenu un bon et beau programme, qui répondait à mes attentes, et à celledu Professeur.

Annexe :

Exemple de fonction d'interfaces :

```
void gotoligcol( int lig, int col ){
    COORD mycoord;

    mycoord.X = col;
    mycoord.Y = lig;
    SetConsoleCursorPosition( GetStdHandle( STD_OUTPUT_HANDLE ), mycoord );
}

void centre(char * string){ //centrer un chaine de caractère au milieu de la console (modifier LARGEURF_CONS si la console est plus petite
    int i = 0;
    printf("\n");
    while (i< (LARGEUR_CONS - strlen(string)) / 2){
        printf(" ");
        i++;
    }
    printf("%s",string);
}

void bas_droite(char * string){ //mettre un élément dans le coin a droite
    int i = 0;
    gotoligcol(HAUTEUR_CONS-2,(LARGEUR_CONS - strlen(string)-3));
    printf("%s",string);
}
```

Exemples de fonctions d'affichage :

```
void AfficherCitoyens(Tcitoyen * pcitoyens, Tcitoyen * pdebutcitoyens){
    system("cls");
    centre("Affichage de la liste des citoyens : \n\n\n\n");
    pcitoyens = pdebutcitoyens->pnext;

    while(pcitoyens->pnext != NULL ){
        printf(" - %s %s\n",pcitoyens->nom,pcitoyens->prenom);
        pcitoyens = pcitoyens->pnext;
    }
    bas_droite("Press Enter");
}

void AfficherLieux(Tlieu * plieux, Tlieu * pdebutlieux){

void AfficherEvenements (Tlieu * plieux, Tlieu * pdebutlieux){
    system("cls");
    centre("Affichage des evenements\n\n\n");
    plieux = pdebutlieux->pavant;
    while(plieux->pavant != NULL ){
        printf("A %s :\n",plieux->nomLieu);
        if(plieux->pliste != NULL){
            Tparticipant * pdeplacement = plieux->pliste;
            printf(" Le %d :\n",pdeplacement->date);
            int date = pdeplacement->date;
            printf("  - %s %s\n",pdeplacement->ppersonne->nom, pdeplacement->ppersonne->prenom);
            while(pdeplacement->psuivant != NULL){
                pdeplacement = pdeplacement->psuivant;
                if (date != pdeplacement->date)printf(" Le %d :\n",pdeplacement->date);
                printf("  - %s %s\n",pdeplacement->ppersonne->nom, pdeplacement->ppersonne->prenom);
            }
        }
        printf("\n\n");
        plieux = plieux->pavant;
    }
}
```

Exemples de fonction d'ajout :

```
void AjoutCitoyen(Tcitoyen * pcitoyens, Tcitoyen * pdebutcitoyens){
    system("cls");
    centre("Ajout d'un citoyen");
    pcitoyens = pdebutcitoyens;

    while( pcitoyens->pnext->pnext != NULL){
        pcitoyens = pcitoyens->pnext;
    }

    pcitoyens->pnext->pprevious = (Tcitoyen *)malloc(sizeof(Tcitoyen));
    pcitoyens->pnext->pprevious->pnext = pcitoyens->pnext;
    pcitoyens->pnext->pprevious->pprevious = pcitoyens;
    pcitoyens->pnext = pcitoyens->pnext->pprevious;

    pcitoyens = pcitoyens->pnext;

    gotoligcol(10,10);printf("Nom : ");
    scanf("%s",pcitoyens->nom);
    gotoligcol(12,10);printf("Prenom : ");
    scanf("%s",pcitoyens->prenom);
}
```

Exemple de fonction de gestion des fichiers situations :

- concaténation d'un chemin de fichier:

```
//concatenations pour creer le chemin en fonction de ce que l'utilisateur a entre
char pathcitoyen[100]="Fichiers_Situations/", pathlieu[100]="Fichiers_Situations/", pathparticipants[100]="Fichiers_Situations/";
strcat(pathcitoyen,profil);strcat(pathcitoyen,"/Citoyens.dat");
strcat(pathlieu,profil);strcat(pathlieu,"/Lieux.dat");
strcat(pathparticipants,profil);strcat(pathparticipants,"/Participants.dat");
```

- ouverture du fichier situations lieux :

```
//fichier situation Lieux
FILE * fLieux = fopen(pathlieu,"r");
if (fLieux == NULL){gotoligcol(14,74);printf("ABS");}
else {
    //remise a zero de la liste lieux
    pdebutlieux->pprecedent = NULL;
    pfinlieux->pavant = NULL;
    pdebutlieux->pavant = pfinlieux;
    pfinlieux->pprecedent = pdebutlieux;

    char string [40];
    fscanf(fLieux, "%s",string);
    while(string[0] != '$'){
        //insertion d'un maillon
        plieux = pdebutlieux;
        while( plieux->pavant->pavant != NULL) plieux = plieux->pavant;
        plieux->pavant->pprecedent = (Tlieu *)malloc(sizeof(Tlieu));
        plieux->pavant->pprecedent->pavant = plieux->pavant;
        plieux->pavant->pprecedent->pprecedent = plieux;
        plieux->pavant = plieux->pavant->pprecedent;
        plieux = plieux->pavant;
        plieux->pliste = NULL;

        strcpy(plieux->nomLieu,string);
        fscanf(fLieux, "%s",string);
    }
    fclose(fLieux);
    gotoligcol(14,74);printf("OK");
}
```

Fonction de suppression : (affichage et fonction de suppression d'un maillon)

```
void SupprimerApresDate(Tlieu * plieux, Tlieu * pdebutlieux){
    system("cls");
    centre("Suppression des evenements");

    int joursaisi, moissaisi, anneesaisi, jourmaillon, moismaillon, anneemaillon, date;
    gotoligcol(7,2);printf("Choix de la date a partir de laquelle supprimer : ");
    gotoligcol(8,5);printf("Jour :  ");scanf("%d",&joursaisi);
    gotoligcol(9,5);printf("Mois :  ");scanf("%d",&moissaisi);
    gotoligcol(10,5);printf("Annee : ");scanf("%d",&anneesaisi);

    void SupprMaillon(Tparticipant * pdeplacement, Tlieu * plieux){
        pdeplacement = plieux->pliste;
        if(pdeplacement->date == date && pdeplacement->psuivant == NULL)plieux->pliste = NULL;
        else if(pdeplacement->date == date)plieux->pliste = pdeplacement->psuivant;
        else{
            while(pdeplacement->psuivant->date != date)pdeplacement=pdeplacement->psuivant;
            pdeplacement->psuivant = pdeplacement->psuivant->psuivant;
        }
    }
```

WaitPress() et Appui() :

```
void Appui(){ //renvoie la touche appuye ou un combo de deux touches pour les fleches directionnelles
    int c = getch();
    printf("%d",c);
    int c2 = getch();
    printf("%d",c2);
}

void WaitPress(){ //attends la pression de enter ou esc
    int c;
    while ((c != 13) && (c != 27)){c=getch();}
}
```