NF05 – Introduction au langage C

Projet – Gestion des passagers dans un aéroport

Magnier Louis

PELLOUARD Mathieu

Branche : Tronc Commun

Semestre : Automne 2019

Résumé

L'objectif du projet est de développer un logiciel permettant de gérer les passagers dans un aéroport. Ce logiciel implémentera quelques fonctionnalités qu'on trouve aujourd'hui. On trouvera notamment des fonctions simulant l'achat de billet, la conception de boardpass, de l'embarquement d'un avion…

Responsable NF05 : Taha ARBAOUI

Chargé de TD : Florian BLANCHERE

Mots clés (CF Thésaurus)

- Projet NF05
- Aéroport
- Passager

Table des matières

Introduction

L'objectif du projet est de développer un logiciel permettant de gérer les passagers dans un aéroport. Ce logiciel implémentera quelques fonctionnalités qu'on trouve aujourd'hui dans l'aéroport et qui sont définies ci-après.

Le parcours d'un passager à prendre en compte est le suivant : le passager s'enregistre et dépose son bagage. Après le dépôt du ou des bagages, il passe les frontières, la sécurité et il embarque. Deux types de passagers sont à prendre en compte : les passagers sans privilège et les passagers priority. Les passagers priority peuvent déposer deux bagages alors que les passagers sans privilège ne peuvent déposer qu'un seul au maximum. Le bagage d'un passager priority devra aussi être en mention priority.

Le logiciel doit inclure les fonctionnalités suivantes :

1. Création de vol
2. Ajout d'un passager sur un vol (cette étape est pour simuler l'achat de billet)
3. Création d'un boarpass (en accord avec les informations saisies lors de l'achat du billet)
4. Enregistrement des bagages
5. Passage du passager par la frontière
6. Passage du passager par la sécurité
7. Embarquement d'un vol et chargement des bagages dans l'avion
8. Décollage du vol

Le problème principal de ce projet sera donc de gérer les données des passagers depuis leur entrée dans l'aéroport jusqu'au décollage de leur avion.

Ce rapport a pour objectif de présenter notre démarche pour résoudre les problèmes cités ci-dessus tout en donnant des informations relatives au programme comme son mode d'emploi. Pour cela, nous expliquerons tout d'abord le fonctionnement de l'algorithme. Ensuite, nous présenterons les problèmes rencontrés et les solutions adoptées. Enfin, nous donnerons un mode d'emploi du programme puis des perspectives d'amélioration de ce dernier. Il est à noter que le code commenté sera fourni en annexe.

Nous souhaitons également remercier nos enseignants : Taha Arbaoui, Florian Blachère et Rémi Cogranne pour leur pédagogie tout au long de ce semestre.

I. Fonctionnement de l'algorithmes :

I.1. Les structures :

La première étape dans la conception de notre programme a été la création des structures grâce à typedef struct. Finalement, nous avons mis en place quatre structures principales :

1. **Billet** – Elle sera nécessaire notamment pour la fonction achat_billet et contient des informations relatives au passager (prénom, nom, date de naissance, numéro de passeport, destination, nationalité et numéro de billet).
2. **Boardpass** – Cette structure contient toutes les informations présentées dans billet avec des précisions supplémentaires comme le nombre de bagage, le numéro de la place, mais aussi son privilège (indique si oui ou non le passager est considéré priority).
3. **Bagage** – Elle contient toutes les informations relatives a un bagage donc le nom, prénom, numéro de billet, numéro de passeport du propriétaire du bagage. De plus, chaque bagage est numéroté. Pour un passager normal, son numéro de bagage sera toujours 1 (puisqu'il ne possède qu'un bagage). En revanche, pour un passager priority qui peut avoir plusieurs bagages, la numérotation peut varier. Par exemple, le second bagage enregistré d'un passager priority sera numéroté deux.
4. **Vol** – Cette structure contient les informations principales relatives à un vol. On y trouve le numéro du vol, le nombre de place total dans l'avion et la destination du vol.

I.2. Les fichiers :

Pour que l'utilisateur ne soit pas obligé de recréer à chaque fois des vols et des passagers après l'arrêt du programme, nous avons créé quelques fichiers. Puisque que ces fichiers ne sont pas destinés à être lu par l'utilisateur, nous avons opté pour des fichiers binaires qui permettent de sauvegarder des tableaux, des structures de données (ou autres) par une lecture directe sans passer par un parsing du texte. Finalement, nous avons créé trois fichiers principaux :

1. **vol.bin** – Ce fichier rassemble toutes les structures vol.
2. **<numero de vol>.bin** – Pour chaque vol créé, nous concevons un fichier dont le nom correspond au numéro de vol suivi de son extension (.bin). Ce fichier commence par un entier qui indique le nombre de place restant dans l'avion avant que le vol soit considéré complet. Ensuite, nous y mettons le boardpass de chaque passager du vol. Enfin, on retrouve tous les structures bagage associées aux passagers du vol.
3. **passager.bin** – Ce fichier rassemble toutes les structures boardpass.

I.3. Le main : int main( )

Le main est dans notre algorithme comme une sorte de menu qui permet d'accéder au différentes fonctions réalisés pour ce projet. Afin que l'utilisation ne se résume pas à une fonction, nous avons utiliser une boucle indéterminée while. Ainsi, l'utilisateur ne peut sortir du programme qu'en saisissant la caractère 'Q' ou 'q'.

I.4. La création de vol : void creer_vol( )

Cette fonction permet d'enregistrer un vol dans le programme. Pour cela, on remplit d'abord la variable ajout de type vol. Cette structure est ensuite sauvegardée dans le fichier « vol.bin » grâce au pointeur sur FILE, ptr. De plus, les données présentes dans la structure nous servent également à créer un second fichier du type <numero de vol>.bin (créé via ptr2) dans lequel on enregistre le nombre de place disponible dans l'avion et des structures boardpass vierge pour faciliter l'entrée des données des passagers plus tard dans le programme.

I.5. L'achat de billet, enregistrement des bagages et création de boardpass:

L'achat de billet est assurée par la fonction achat_billet( ). La fonction creation_boardpass(billet passager) permet la création du boardpass tandis que enregistrement_bagage (boardpass info,char nom[20],int i) garantit l'enregistrement des bagages.

Dans ces trois fonctions (qui composent le bloc « Achat Billet » dans le programme), le but est de remplir les structures de données billet, boardpass et bagage. Les algorithmes sont assez simples et sont majoritairement composés de demandes d'informations. Pour les données qui présentent un risque d'erreur d'entrée pour l'utilisateur, on effectue quelques contrôles assez basiques pour s'assurer d'avoir des informations utilisables pour la suite du programme. Toutes ses données sont stockées dans des fichiers dans le but de pouvoir les conservées même dans le cas où l'on termine l'exécution du programme pour reprendre plus tard. Ainsi, le boardpass créé est écrit dans le fichier <numéro de vol>.bin et dans passager.bin.

I.6. Passage de la frontière : void frontieres ( )

Cette fonction permet de faire passer l'ensemble des passagers enregistrés dans passager.bin par la frontière. On affichera sa destination, sa nationalité et s'il a besoin d'un visa. Pour ce faire, nous allons lire les boardpass stockés dans passager.bin un par un jusqu'à la fin du fichier grâce à fichier (qui est un pointeur sur FILE) et une boucle while. Le boardpass lu sera stocké provisoirement dans passager qui est de type boardpass. Ensuite nous avons distingué plusieurs cas : si le passager est américain, si le passager est « européen » (toutes les nationalités européennes ne sont pas prises en compte), si le passager est chinois et si le passager a une nationalité différente de celles citées au-dessus. Pour les 3 premiers cas, on considère que le passager a besoin d'un visa quand sa destination diffère de son pays d'origine. Par exemple, pour le cas américain, si la destination du passager est les USA alors ce dernier n'a pas besoin d'un visa dans tous les autres cas, il en a besoin. Pour les autres nationalités, le programme n'affiche aucune information.

I.7. Passage de la sécurité : void securite( )

Cette fonction permet de faire passer un passager lambda par la sécurité. A la sécurité, les bagages à main sont passés au scanner. C'est ce que l'on propose de simuler dans cette fonction. Ici, l'utilisateur joue le rôle du scanner et doit répondre à certaines questions. Tout d'abord, on demande à l'utilisateur de saisir le nombre d'affaires dans le bagage à main. Sa réponse sera stockée dans nb_affaire. Pour que cette fonction donne le résultat attendu, il faut que l'utilisateur respecte certaines règles de saisie qu'on affiche via un printf. Ensuite, on entre dans une boucle for qui commence a 0 et s'arrête à nb_affaire-1. Puis l'on demande le nom de la i-ème affaire qui sera enregisté dans affaire. Selon le mot saisi, on affiche alors s'il faut ou non retirer le produit. Là encore, nous n'avons pas traité tous les cas. Pour en traiter le plus que possible, nous avons réuni certain objets en grande catégorie que nous affichons avant de demander à l'utilisateur de saisir le nom de l'objet « scanné ». Par exemple, au lieu de marquer fusil l'utilisateur devra saisir arme_a_feu.

I.8. L'embarquement : boardpass* embarquement( )

I.8.1. Principe de la fonction :

Cette fonction permet la simulation d'un embarquement. Ici, le but est de simplement copier les passagers du vol que l'utilisateur souhaite faire embarquer dans un tableau dynamique. Les passagers sont transférés en fonction de leur privilège : les passagers « priority » seront embarqués avant les passagers « normaux ». A la fin de cette fonction l'utilisateur a deux choix : soit l'avion décolle tout de suite auquel cas les données sont transférées dans la fonction décollage, soit le décollage est reporté à plus tard et dans ce cas, les données seront réenregistrées dans deux fichiers créés à l'occasion.

I.8.2. Fonctionnement étape par étape :

Etape 1 : On doit afficher les vols qu'y peuvent être embarquer (c'est-à-dire les vols dont l'ensemble des places est vendu). Pour cela on ouvre le fichier vol.bin grâce a ptr et on lit une par une les structures vol. En réalité, on s'intéresse uniquement au numero de vol de chaque avion que l'on copie dans nom (qui est une chaine de caractère). On rajoute alors l'extension .bin puis l'on ouvre le fichier <numero de vol>.bin via ptr2. On enregistre alors le nombre de place disponible dans l'avion dans retenu. Si retenu=0, cela signifie que toutes les places ont été vendu donc que l'avion peut être embarqué. On affiche donc le numéro de vol et on incrémente i qui correspond au nombre d'avion embarquable. On répète ces étapes jusqu'à arriver à la fin du fichier vol.bin.

Etape 2 : On distingue ensuite deux cas. Soit il n'y a aucun avion embarquable (ce qui se traduit dans le code par i=0) soit il y en a un ou plusieurs. Dans le premier cas, on explique à l'utilisateur qu'aucun avion est embarquable puis on le renvoie vers le main, dans le second, on demande à l'utilisateur quel avion embarqué.

Etape 3 : Ensuite on recherche le nombre de place total dans l'avion pour connaitre la taille du tableau dynamique passage (de type boardpass) qu'on va devoir créer. Pour cela, on utilisera les informations contenus dans le fichier vol.bin. Enfin, on crée le tableau dynamique.

Etape 4 : Ensuite, on remplit le tableau passage (pour chaque boardpass rajouté dans le tableau, on considère le passager correspondant embarqué). On se déplace une première fois dans le fichier <numéro de vol>.bin pour trouver les passagers « priority ». Dès qu'on en trouve un, on le copie dans notre tableau dynamique passage. A la fin de notre boucle, on a ainsi, copier tous les boardpass des passagers priority. On réalise cette opération une deuxième fois, mais cette fois-ci, on recherche et on copie les passagers « normaux ». Quand l'opération est terminée, on affiche notre tableau (seulement le nom et prénom). L'embarquement des passagers est alors terminé. On doit procéder à l'embarquement des bagages.

Etape 5 : On se déplace dans notre fichier <numéro de vol>.bin pour accéder aux informations relatives aux bagages. Là encore, on doit d'abord déterminer le nombre de bagage que l'on devra embarquer dans l'avion pour pouvoir ensuite créer un tableau dynamique de type bagage. Quand on a déterminé le nombre de bagage a embarqué (stocké dans i), on crée le tableau dynamique tab.

Etape 6 : Ensuite, on effectue la copie des bagages du fichier <numéro de vol>.bin dans tab.

Etape 7 : Après l'étape 6, l'avion est embarqué. Comme dit précédemment, on distingue deux cas après l'embarquement. Soit l'utilisateur souhaite faire décoller l'avion tout de suite dans ce cas, on appelle la fonction décollage en envoyant notamment les tableaux dynamiques créés. Soit l'utilisateur ne fait pas décoller l'avion tout de suite. Dans ce cas, il faut pouvoir retrouver nos tableaux plus tard. Pour cela, on crée deux nouveaux types de fichiers : E<numéro de vol>.bin et BE<numéro de vol>.bin qui contiennent respectivement les boardpass des passagers embarqués et les bagages embarqués.

I.9. Le décollage : void decollage(...)

I.9.1. Principe de la fonction

Cette fonction simule le décollage d'un vol. L'objectif est de comparer le nombre de passager embarqué avec le nombre de place total de l'avion. On a supposé que nos avions sont embarqués uniquement quand toutes les places ont été vendus. Ainsi, si les deux nombres sont égaux, cela signifie que tous les passagers sont montés à bord. En revanche, s'ils sont différents, cela impliquerait le non-embarquement d'un des passagers. En réalité, le deuxième cas n'est jamais atteint puisqu'à la fin de fonction embarquement tous les passagers ont embarqué. On réalise ensuite une deuxième vérification pour déterminer le nombre total de bagage que doit contenir l'avion et le comparer avec le nombre total de bagage embarqué. Si ces deux nombres sont égaux, l'avion peut décoller. Là encore, le cas où les nombres sont différents ne se produit jamais dans notre programme. Après vérification, on simule le décollage de l'avion. Dans notre programme cela se traduit par la suppression des informations relatives au vol et à ses passagers.

Il faut noter également que la fonction décollage peut être appelé dans deux situations différentes. Soit on l'appelle tout de suite après la fonction embarquement, dans ce cas, on utilisera énormément les paramètres d'entré pour faire les vérifications ; soit, la fonction decollage est appelée via le main, dans ce cas, on doit demander quel vol faire décoller puis l'on utilise des fichiers pour s'assurer que le vol puisse décoller.

I.9.2. La fonction étape par étape :

Nous devons vérifier que tous les passagers et tous les bagages sont embarqués dans l'avion. On prendra d'abord le cas où l'utilisateur appelle la fonction via le main.

Etape 1 : On affiche les vols qui ont été embarqués préalablement. Pour cela, on ouvre le fichier vol.bin grâce au pointeur sur FILE, fichier puis l'on lit un par un les numéro de vol de chaque vol enregistré dans vol.bin. Cette information est stockée dans navig (de type vol) puis on essaie d'ouvrir le fichier E<numéro de vol>.bin (fichier qui contient tous les passagers embarqués pour le numéro de vol correspondant). Si le fichier existe, cela indique que l'embarquement du vol en question est terminé, on peut donc proposer le décollage de cette avion via un printf puis on incrémente le nombre d'avion embarqué (nb_avion_emb). On distingue deux possibilités. Il est possible qu'aucun vol ne puisse décoller (car aucun embarquement a été réalisé préalablement) dans ce cas, on l'indique via un printf et l'on affecte à la variable pret la valeur 0. Sinon, il y a un ou plusieurs vols qui peuvent décoller dans ce cas, on laisse l'utilisateur choisir le vol qu'il préfère décoller et on affecte la valeur 1 a prêt.

Etape 2 : Si des vols sont embarqués, on détermine tout d'abord le nombre de place total dans l'avion en cherchant l'information dans le fichier vol.bin. On stocke d'abord l'information dans navig puis dans nb_place qui est un entier.

Etape 3 : On commence par ouvrir le fichier E<numéro de vol>.bin via efichier (pointeur sur FILE) puis on incrémente nb_passager_emb (int) qui correspond au nombre de passager embarqué. Si le nombre de place total correspond au nombre de passagers embarqué, on affiche « Tous les passagers ont embarqué ». Il faut maintenant vérifier qu'on a embaqué tous les bagages.

Etape 4 : Il faut déterminer le nombre de bagage que doit contenir l'avion théoriquement avant de décoller. Cette information n'est contenue dans aucun fichier, il faut la trouver intuitivement grâce aux données enregistrées dans <numéro de vol>.bin. On se déplace pour accéder à la dernière partie de notre fichier contenant uniquement des structures bagages. Ensuite, pour chaque structure lu, on incrémente un compteur nb_bag. Il n'y a pas de donnés enregistrés après la partie « bagage », on peut donc lire jusqu'à la fin du fichier.

Etape 5 : De manière analogue à l'étape 3, on trouve le nombre de bagage embarqué grâce au fichier BE<numéro de vol>.bin. On conserve la valeur trouvée dans nb_bag_emb (int). Si le nombre de bagage théorique après l'embarquement correspond au nombre de bagage embarqué, on affiche « Tous les bagages ont été embarqué ».

Les étapes 1 à 5 permettent de vérifier que l'avion peut décoller dans le cas où l'utilisateur appelle la fonction via le main. Dans le cas où l'on décolle l'avion directement après l'embarquement, nous n'avons pas besoin d'afficher tous les vols embarqués, ni de déterminer le nombre de passager et de bagage embarqué. Pour trouver le nombre de place total dans l'avion et le nombre de bagage théorique après embarquement, on suit la même logique que dans les étapes précédentes. Remarque, pour ce cas, on affecte toujours la valeur 1 a la variable prêt.

Etape 6 : La première chose à vérifier est que prêt=1, si ‘est e cas, cela signifie qu’un avion peut décoller. Si prêt=0, aucun avion n’a été embarqué donc aucun ne peut décoller.

Etape 7 : Si un vol est prêt pour le décollage, nous devons supprimer les informations concernant ses passagers. Pour chaque passager embarqué, on réalise l’opération suivante :

- On crée un nouveau fichier binaire qu’on nomme newpassager.bin via le pointeur fnew.
- On lit les structures boardpass contenues dans passager.bin une par une et on la stocke dans pass_navig (de type boardpass). Vu que les numéro de passeport sont nominatif, on compare le numéro de passeport de notre passager embarqué avec le numéro de passeport qu’on a stocké dans pass_navig. S’ils sont différents, cela signifie que la personne que nous venons de lire dans passager.bin n’est pas notre passager embarqué donc on peut la copier dans newpassager.bin. Par contre, si les numéro de passeport sont identiques dans ce cas la personne lue dans passager.bin correspond au passager embarqué. Dans ce cas, cette personne ne doit pas être copier dans newpassager.bin.
- On ferme nos deux fichiers puis l’on supprime le fichier passager.bin.
- On renomme le fichier newpassager.bin par passager.bin.

Ainsi, à la fin de la première itération, nous avons supprimé le 1er passager embarqué de notre fichiers passager.bin. On doit donc réaliser cette étape autant de fois qu’il y a de passager embarqué dans l’avion qui s’apprête à décoller. Si nous sommes dans le cas où l’avion décolle juste après la fonction embarquement, on utilisera notre tableau dynamique tabpass. Sinon, on utilisera notre fichier E<numéro de vol>.bin puis on lira un par un chaque passager embarqué. On stockera l’information dans pass_emb de type boardpass.

Etape 8 : Nous devons également supprimer l’avion qui est sur le point de décoller dans le fichier vol.bin. Pour cela, on suit la même logique que dans l’étape précédente. On crée newvol.bin avec fnew, on lit dans le fichier vol.bin chaque structure vol que l’on enregistre dans navig. Si le numéro de vol contenu dans navig et le numéro de vol de l’avion qui décolle sont différent alors on copie le vol dans newvol.bin sinon cela signifie qu’on est tombé sur l’avion qui va décoller donc on n’écrit pas dans navig dans le nouveau fichier. Quand l’opération est terminée, on supprime vol. et on renomme newvol.bin, vol.bin. Ainsi, le vol qui va décoller est supprimer de notre fichier vol.bin.

Etape 9 : La dernière consiste à supprimer le <numéro de vol>.bin. Si l’utilisateur a appelé la fonction décoller via le main, cela signifie qu’il existe un E<numéro de vol>.bin et un BE<numéro de vol>.bin. Dans ce cas, nous devons également les supprimer. Quand la ou les suppression(s) sont terminée(s), nous n’avons plus aucune information sur le vol qui devait décoller. On indique donc à l’utilisateur que le décollage est terminé.

II. Problèmes rencontrés et solution adaptés :

La gestion de passager et de vol nécessite un stockage des données particuliers permettant de récupérer des informations facilement. Cela sert notamment au traitement des demandes de l'utilisateur mais également au suivi des passagers et des vols. La méthode de stockage dont nous avions besoin devait être capable de conserver les informations entre plusieurs exécutions du programme pour éviter de devoir ré-entrer des données à chaque exécution. Pour ce faire nous avons donc décidé d'utiliser des fichiers binaires.

Les principaux problèmes que nous avons rencontrés sont liés à la lecture et l'écriture de nos fichiers. Nous avons mis un certain temps avant de comprendre parfaitement le fonctionnement de ces derniers. L'écriture au début du projet nous a posé un problème puisque l'organisation au sein de nos fichiers était encore floue. Quand nous avons trouvé une répartition d'informations qui nous convenait, il a fallu qu'on se tienne à cette organisation. Cependant, l'écriture dans les fichiers demande une rigueur essentiellement les « numéro de vol ».bin puisqu'elle possède plusieurs types de variables (int puis boardpass et enfin bagage). Pour faciliter l'écriture, nous avons opté pour la solution suivante qui consiste à écrire des structures vierges dans le fichier. Ainsi, on a pu modifier ce dernier à certains emplacements sans avoir à le lire complètement, le stocker de manière dynamique et le réécrire avec l'informations supplémentaires. La lecture quant à elle nous a perturbé à cause de multiple dépassement de mémoire que nous n'avions pas spécialement compris.

Un second problème que nous avons rencontré est celui de la pertinence du programme. Pour s'approcher le plus possible de la réalité, nous avons fait un certain nombre de recherches par exemple pour les formats de numéro de passeport ou les passages de sécurité.

Un autre problème a été de rendre le programme lisible et pour cela nous avons décidé de réinitialiser la console au début de certaines fonctions. Pour cela, on utilise la fonction system(« CLS »). Un autre problème au niveau de la lisibilité du programme a été l'encodage des caractères spéciaux type « é », « à » ou encore « è ». Pour cela nous avons simplement décidé de remplacer chacune de ces lettres par un « %c » puis nous mettons le numéro du caractère correspondant dans la table ASCII.

Enfin, pour les fonctions frontiere( ) et securite( ), un trop grand nombre de cas devaient être pris en compte. Pour traiter le maximum de cas, nous avons simplifier le problème.

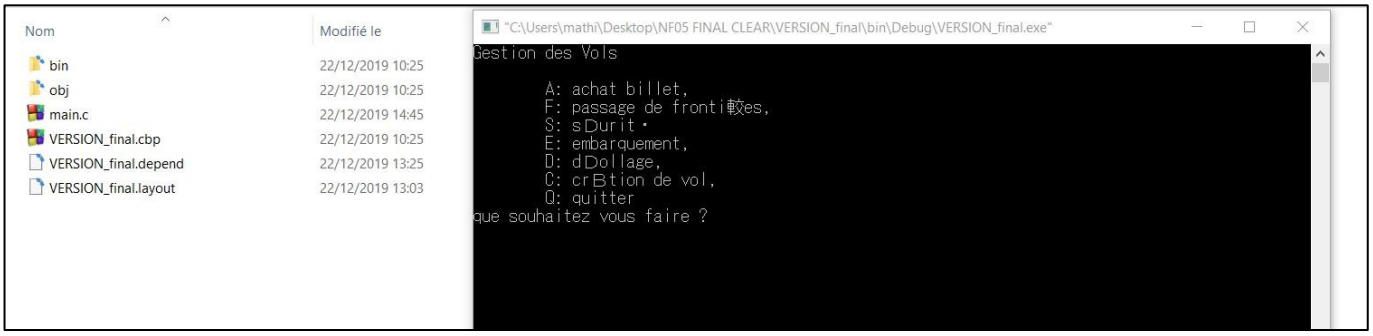
- Dans frontiere( ), si le passager a une nationalité reconnue par le programme (FRANCAIS, AMERICAIN, ANGLAIS, ALLEMAND, ESPAGNOL, CHINOIS), alors si la destination correspond au pays d'origine du passager, celui-ci n'a pas besoin de visa. Pour toutes les nationalités européennes, si la destination est FRANCE, ANGLETERRE, ALLEMAGNE, ESPAGNE alors le passager n'aura pas besoin de visa non plus. Ensuite, si la nationalité est reconnue par le programme mais pas la destination, alors on indique forcément que le passager a besoin d'un visa. Enfin, si la nationalité n'est pas reconnue, nous n'indiquons aucune information par rapport au visa.
- Dans securite( ), nous avons créer une longue liste d'objet interdis en cabine qui est non-exhaustif. Pour traiter le maximum de cas, certains ont été réunis en grande famille qui sont donné préalablement à l'utilisateur (exemple : arme_a_feu, alcool, produit_menager,…)

III. Mode d'emploi du programme et exemple d'utilisation :

Règle de saisie : A moins que cela soit précisé lors de l'affichage, il est préférable de saisir tout en majuscule et de remplacer les espaces par des underscores (attention : la fonction sécurité demande d'écrire en minuscule).

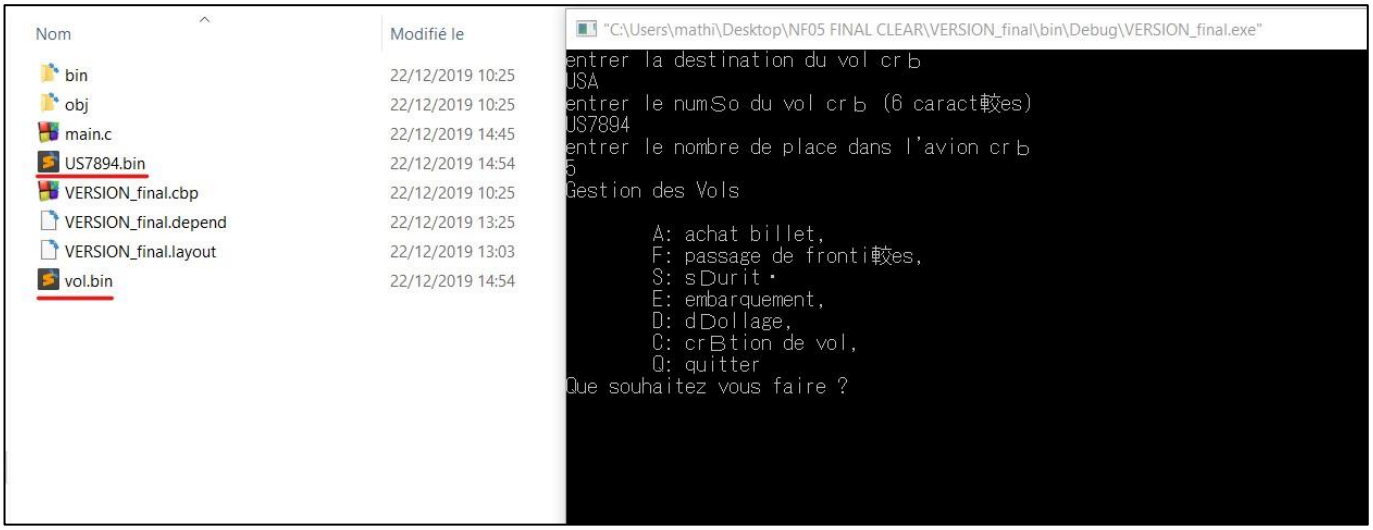
Exemple du programme et conseil d'utilisation :

Quand on lance le programme, l'utilisateur arrive dans un menu (le main) qui permet d'accéder au différentes fonctions demandées pour le projet. Pour utiliser le menu, vous devez saisir le caractère placé devant la fonction choisi (minuscule ou majuscule fonctionne pour ce cas). L'image ci-dessous présente le menu et le dossier lorsqu'on utilise le programme pour la première fois.



Il n'y a pas réellement d'ordre d'utilisation pour les fonctions. Néanmoins, il est conseillé lorsqu'on utilise le programme pour la première fois (ou lorsqu'on a aucun fichier dans le dossier), de suivre un certain ordre d'utilisation. Voici les étapes qu'on vous conseille de suivre (à partir du menu) si vous êtes dans ce cas :

Etape 1 : Choisir la fonction « création de vol » en saisissant 'C' dans le menu. Suivez ensuite les instructions affichées à l'écran. Un exemple de saisie pour l'option « création de vol » est donné ci-contre. On remarquera la création de vol.bin et d'un fichier du type <numéro de vol>.bin (souligné en rouge dans l'image).



Remarque : noter bien la destination en majuscule

Etape 2 : Lorsqu'un vol est créé, on revient vers le menu. Il faut maintenant que des personnes achètent les places de l'avion « mise en vente ». La deuxième fonction qu'on doit appeler est donc « achat de billet ». Pour cela, il faut saisir 'A' dans le menu puis suivre les instructions. L'image suivante présente une saisie réalisé dans l'option « achat billet ».

```
Achat Billet

entrer votre prOom
Mathieu
entrer votre nom
PELLOUARD
entrer votre date de naissance
entrer le jour : 12
entrer le mois : 07
entrer l'annF : 2000
entrez votre numSo de passeport(2 chiffres puis 2 lettres et 5 chiffres)
78AZ78945
les destinations disponibles sont les suivantes :
       USA
entrer votre destination :
USA
 votre numFro de billet et le suivant : NLUS7894PM
entrer votre nationalit・au masculin, en majuscule et sans caract較e spDiaux
FRANCAIS
boardpass en crBtion
quel privil拡e souhaitez vous avoir : normal:0 ou priority:1
1
combien de bagage souhaitez vous emportez ?
2
bagage enregistr・
bagage enregistr・
dTirez-vous choisir votre place : si oui tapez 1 si non tapez 0
1
vous pouvez choisir une place entre 0 et 4
3
Votre place sera donc 3
```

Remarque : Noter bien la nationalité en majuscule

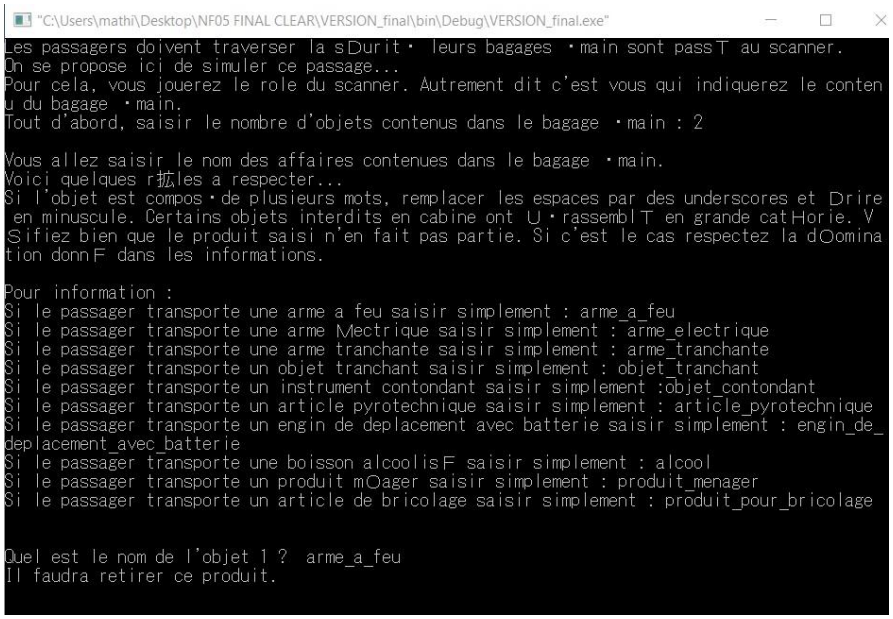
On remarquera la création d'un fichier passager.bin dans le dossier de notre programme.

Nom	Modifié le	Type	Taille
bin	22/12/2019 10:25	Dossier de fichiers	
obj	22/12/2019 10:25	Dossier de fichiers	
main.c	22/12/2019 15:05	Fichier C	35 Ko
passager.bin	22/12/2019 15:06	Fichier BIN	1 Ko
US7894.bin	22/12/2019 15:06	Fichier BIN	3 Ko
VERSION_final.cbp	22/12/2019 10:25	Fichier CBP	2 Ko
VERSION_final.depend	22/12/2019 13:25	Fichier DEPEND	1 Ko
VERSION_final.layout	22/12/2019 13:03	Fichier LAYOUT	1 Ko
vol.bin	22/12/2019 14:54	Fichier BIN	1 Ko

Pour pouvoir embarquer, il faut que toutes les places de l'avion soient achetées. Dans l'exemple donné, cela signifie que l'utilisateur doit utiliser la fonction achat de billet 5 fois (car il y a 5 places dans l'avion) s'il veut faire embarquer l'avion.

Etape 3 : De retour au menu, on vous conseille ensuite de choisir la fonction « passage de frontière ». En effet, si on suit le parcours d'un passager dans un aéroport, après l'enregistrement des bagages, ce dernier passe les frontières. Pour cela, saisir 'F' dans le menu. Dans cette fonction, il ne faut rien saisir. Le programme affiche uniquement la nationalité du passager, sa destination et s'il a besoin d'un visa. Voici ci-dessous un exemple d'affichage obtenue via la fonction « frontière ». On pourra remarquer ici qu'un américain n'a pas besoin de visa car sa destination correspond à son pays d'origine (souligné en rouge sur l'image)

Etape 4 : De retour au menu, on vous conseille maintenant de choisir l'option « Sécurité ». Dans notre exemple, cela signifie que nos passagers se font contrôler par la sécurité. On rappelle ici que la fonction fait passer un passager lambda (on peut donc utiliser cette fonction à l'infini). Là encore, l'ensemble des instructions sont affichées par le programme. Voici un exemple de saisie pour la fonction sécurité.

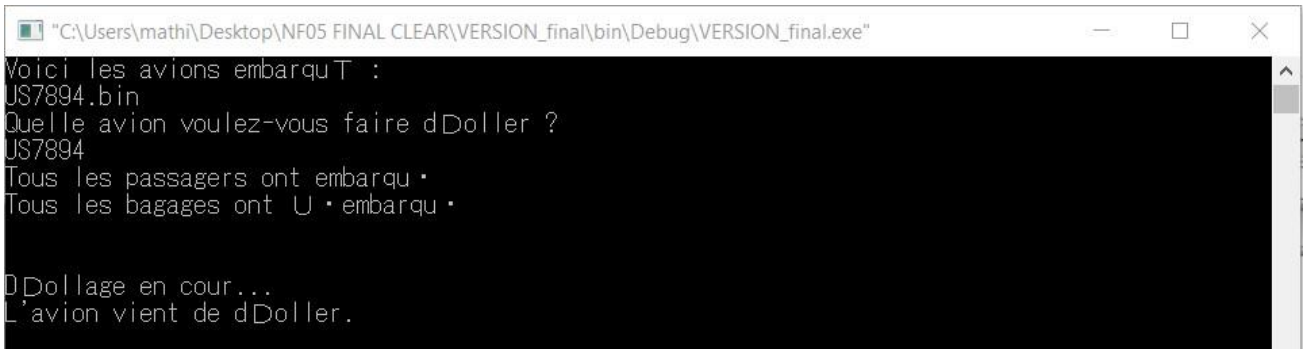


Remarque : Il faut absolument que l'utilisateur écrive en minuscule et remplace les espaces par des underscores.

Etape 5 : On vous conseille maintenant d'utiliser la fonction « embarquement ». Pour cela, saisissez 'E' dans le menu puis suivez les instruction. Attention, le programme va d'abord afficher les numéro de vol que vous pouvez faire embarquer suivi de l'extension .bin. Quand le programme demandera d'entrer le numéro de vol à faire embarquer **ne pas saisir cette extension.** Voici un exemple d'utilisation de la fonction « embarquement » (ici nous avons décidé d'attendre avant le décollage) :

Dans notre exemple, le choix réalisé dans embarquement (où l'on saisit 0) entraine la création de deux fichiers du type E<numéro de vol>.bin et BE<numéro de vol>.bin qui sont souligné en rouge dans l'image suivante.

Etape 6 : Maintenant, vous pouvez faire décoller le vol. Pour cela, vous pouvez passer par la fonction « embarquement » qui vous propose de procéder au décollage du vol. Sinon vous pouvez passé par le menu et taper 'D'. Suivez ensuite les instructions. Là encore, si le programme demande « quel avion voulez-vous faire décoller », il ne faut pas saisir l'extension .bin affichée dans la liste des avions embarqués. Voici, un exemple lorsqu'on utilise la fonction « décollage » :



On notera la suppression des fichiers du type <numéro de vol>.bin, E<numéro de vol>.bin BE<numéro de vol>.bin dans le dossier comme le montre l'image suivante.

Quelques remarques :

- Si des résultats surprenants apparaissent, regardez vos fichiers. Si des fichiers newpassager.bin ou newvol.bin subsistent à la fin de la fonction « décollage », c'est probablement lié à une erreur de saisie dans l'une des fonctions. Dans ce cas, il est conseillé de supprimer tous les fichiers avec l'extension .bin (y compris vol.bin et passager.bin).
- L'ordre donné ci-dessus n'est pas obligatoire. Néanmoins, il est fortement conseillé de le suivre car certaines fonctions n'ont aucun intérêt si l'ordre n'est pas respecté un minimum. Ainsi :
    - La fonction « frontiere » renvoie tout de suite vers le menu (main) s'il n'existe pas de fichier passager.bin ou si ce dernier est vide.
    - La fonction « achat_billet » finit par renvoyer l'utilisateur vers le menu (main), s'il n'existe pas de fichier vol.bin ou si ce dernier est vide.
    - La fonction « embarquement » finit par renvoyer l'utilisateur vers le menu si aucun vol n'est complet.
    - La fonction decollage finit par renvoyer l'utilisateur vers le menu si aucun vol est embarqué.
- Les captures d'écrans réalisées ne sont pas entièrement à jour, il peut donc avoir de légères différences. De plus, l'affichage du texte dans nos captures n'est pas celui attendu (cela est lié au paramètre de l'encodage qui n'est pas le plus courant sur l'ordinateur utilisé).

Conclusion et amélioration possible :

En conclusion, ce projet restera une bonne expérience pour notre groupe. Tout d'abord, au niveau du C, cette fonction nous as permis de construire des bases beaucoup plus solides sur les fichiers. Ensuite, nous avons tous les deux été confrontés au travail de groupe dans le cadre d'un projet de programmation pour la première fois. Nous étions généralement d'accord sur le chemin à suivre pour atteindre les objectifs fixés. En revanche, nous n'avons pas la même méthode de travail et nous ne programmons pas de la même manière. Ainsi, chacun a dû adapter sa méthode de travail pour qu'on aboutisse à un projet commun.

Par rapport à notre projet, certaines améliorations peuvent être apportées. Tout d'abord, nous aurions pu demander quelle rôle l'utilisateur souhaitait jouer dans le programme (passager ou opérateur dans l'aéroport). Selon la réponse de l'utilisateur, notre programme se serait adapté pour ne laisser que quelques fonctions disponibles. Ensuite, on aurait préféré traiter plus de cas possible au niveau des fonctions « frontières » et « sécurité ». Enfin, on aurait dû laisser la possibilité de choisir quelle personne faire passer par la frontière ou la sécurité. Enfin, il aurait été intéressant de réaliser quelques bonus ou de proposer une sorte d'interface graphique…

Annexe :

Référence :

Afin de déterminer les objets interdits en cabine, nous avons utilisé le lien suivant :
https://www.airfrance.fr/FR/fr/common/guidevoyageur/pratique/produits-interdits-et-reglementes-airfrance.htm

Code commenté :

```
/*!
* \file main.c
* \brief ce programme permet la gestion des passagers et des avions pour un aeroport.
*
* \version 1.0
* \date 20/12/2019
* \author Magnier Louis Pellouard Mathieu
*/
/*!
* \mainpage Gestion de passager dans un aeroport
* Ce programme permet de gerer des passagers depuis leurs arrives dans l'aeroport jusqu'a leurs
embarquement
dans l'avion. Le programme gere egalement les differents vols.
*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <ctype.h>
/*!
* \struct billet
* \brief cette structure correspond aux informations contenus sur un billet
*
*
*
*/
typedef struct billet
//Ici, on definit le premier type qui correspond aux informations necessaires pour generer un billet.
{
char prenom[100]; ///cette variable sert à stocker le prenom du passager
char nom[100]; ///cette variable sert à stocker le nom du passager
int naissance[3]; ///cette variable sert à stocker le jour puis le mois et l'année de naissance d'un
passager
char passeport [10]; ///cete variable sert à stocker le numéro de passeport du passager
char dest [100]; ///cette variable sert à stocker la destination du passager
char nationalite[100]; ///cette variable sert à stocker la nationalité du passager
char numbillet[11]; ///cette variable sert à stocker le numéro de billet du passager
}billet;
/*!
* \struct boardpass
* \brief cette structure correspond aux informations contenus dans un boarding pass
*/
typedef struct boardpass
//Ici, on definit le boarding pass (toutes les caracteristiques du passager)
{
char prenom[100]; ///cette variable sert à stocker le prenom du passager
char nom[100]; ///cette variable sert à stocker le nom du passager
int naissance[3]; ///cette variable sert à stocker le jour puis le mois et l'année de naissance d'un
passager
char passeport[10]; ///cete variable sert à stocker le numéro de passeport du passager
char dest [100]; ///cette variable sert à stocker la destination du passager
char nationalite[100]; ///cette variable sert à stocker la nationalité du passager
char numbillet[11]; ///cette variable sert à stocker le numéro de billet du passager
int privilege; ///cette variable sert à stocker le privilège du passager
int nbbagage; ///cette variable sert à stocker le nombre de bagage que le passager emporte
int numplace; ///cette variable sert à stocker le numéro de place du passager
}boardpass;
/*!
* \struct bagage
* \brief cette structure correspond aux informations necessaires pour stocker un bagage
*/
typedef struct bagage
//Ici, on définit le type qui permettra d'identifier chaque bagage
{
char prenom[100]; ///cette variable sert à stocker le prenom du passager
char nom [100]; ///cette variable sert à stocker le nom du passager
char numbillet[11]; ///cette variable sert à stocker le numéro de billet du passager
int numbagage; ///cette variable sert à stocker le numéro du bagage du passager
char passeport[10]; ///cette variable sert à stocker le numéro de passeport du passager
int privilege; ///cette variable sert à stocker le privilege du passager
```

```
}bagage;
/*!
* \struct vol
* \brief cette structure correspond aux informations necessaire pour creer un vol
*/
typedef struct vol
{
char numerovol[6]; ///cette variable sert à stocker le numéro de vol
int nbplace; ///cette variable sert à stocker le nombre de place du vol
char dest[50]; ///cette variable sert à stocker la destination du vol
}vol;
/*!
* \fn void creer_vol()
* \brief Cette fonction permet de créer un vol ainsi que tout ce qu'il implique au niveau de la gestion
des
données.
*
*/
void creer_vol() //cette fonction permet de créer un vol qui pourra être utilisé dans la suite du
programme
{
int i;
boardpass vierge;
FILE *ptr;
FILE *ptr2;
vol* ajout=(vol*)malloc(1*sizeof(vol));
char nom2[20];
//à partir d'ici, on s'occupe du fichier dans lequel sont stockés tout les vols
char nom[20]="vol.bin";
system("CLS");
ptr=fopen(nom ,"a+");
//jusque la ligne 69 on remplit la structure vol qui sera entrée dans le fichier vol.bin et qui servira à
créer le fichier du type «numerodevol ».bin
printf("entrer la destination du vol cr%c%c (ecrire en majuscule et remplacer les espaces par des
underscores)\n",130,130);
scanf("%s",ajout->dest);
printf("entrer le num%cro du vol cr%c%c (6 caract%cres)\n",130,130,130,138);
scanf("%s",ajout->numerovol);
printf("entrer le nombre de place dans l'avion cr%c%c\n",130,130);
scanf("%d",&ajout->nbplace);
strcpy(vierge.nom," ");//ici on remplit une structure qu'on rentrera dans le fichier « numerodevol ».bin
pour faciliter la saisie des données des passagers plus tard
fwrite(ajout, sizeof(vol), 1, ptr);// on ajoute simplement la structure au fichier vol.bin
//sur les deux prochaines lignes on affecte le nom du fichier « numerodevol ».bin
strcpy(nom2,ajout->numerovol);
strcat(nom2,".bin");
ptr2=fopen(nom2,"a+b");//on ouvre ce fichier
// jusque la fin de la fonction,on rentre le nombre de place au debut puis les boardpass vierge (on en
rentre autant qu'il y a de places)
fwrite(&ajout->nbplace,sizeof(int),1,ptr2);
for(i=0;i<ajout->nbplace;i++)
{
fwrite(&vierge,sizeof(boardpass),1,ptr2);
}
fclose(ptr);
fclose(ptr2);
}
/*!
* \fn bagage enregistrement_bagage(boardpass info,char nom[20],int i)
* \brief Cette fonction permet de remplir la structure bagage et de stocker les données dans le fichier
correspondant au vol qu'empreinte le passager.
*/
bagage enregistrement_bagage(boardpass info,char nom[20],int i)
{
//le but de cette fonction est de remplir la structure bagage et de la saisir dans le fichier
"numerodevol".bin correspondant au vol que prend le propriétaire du bagage
bagage passage;
FILE* volnom;
//on commence par remplir la structure
strcpy(info.nom,passage.nom);
strcpy(info.prenom,passage.prenom);
strcpy(passage.numbillet,info.numbillet);
```

```
strcpy(passage.passeport,info.passeport);
passage.privilege=info.privilege;
passage.numbagage=i;
//on saisie la structure à la fin du fichier "numerodevol".bin
volnom=fopen(nom,"ab");
fwrite(&passage,sizeof(bagage),1,volnom);
fclose(volnom);
printf("bagage enregistr%c\n",130);
return passage;
}
/*!
* \fn boardpass creation_boardpass (billet passager)
* \brief Cette fonction permet de remplir la structure boardpass et de stocker les données dans le fichier
correspondant au vol qu'empreinte le passager.
*/
boardpass creation_boardpass (billet passager)
{
//le but ici va etre de créer le boardpass. Ici on rentrera toutes les informations.
boardpass passage;
int i, choix;
int* retenu=(int*)malloc(sizeof(int));
FILE* ptr;
FILE* ptr2;
FILE* ptr3;
boardpass navig;
char nom[20]="vol.bin";
char nom2[20];
vol transit;
//on remplit la structure boardpass
//les lignes 6 prochaines lignes sont des copies de la structure billet remplit dans la fonction du meme
nom.
printf ("boardpass en cr%cation\n",130);
strcpy(passage.prenom,passager.prenom);
strcpy(passage.nom,passager.nom);
strcpy(passage.dest,passager.dest);
strcpy(passage.nationalite,passager.nationalite);
strcpy(passage.passeport,passager.passeport);
strcpy(passage.numbillet,passager.numbillet);
//on demande la date de naissance avec une simple boucle for
for (i=0;i<3;i++)
{
passage.naissance[i]=passager.naissance[i];
}
// de la ligne 3 prichaines lignes on collecte encore des informations de l'utilisateur pour la structure
et
pour les bagages.
printf("quel privil%cge souhaitez vous avoir : normal:0 ou priority:1\n",138);
scanf("%d",&passage.privilege);
printf("combien de bagage souhaitez vous emporter ?\n");
scanf("%d",&passage.nbbagage);
//à partir d'ici, on va rechercher dans le fichier "vol.bin" le numero de vol à partir de la destination
saisie par l'utilisateur.
ptr=fopen(nom,"rb");
rewind(ptr);
fread(&transit,sizeof(vol),1,ptr);
while (strcmp(transit.dest,passage.dest)!=0)
{
fread(&transit,sizeof(vol),1,ptr);
}
fclose(ptr);
//on va finir de preparer le nom du fichier "numerodevol".bin et on l'enverra dans la fonction
enregistrement_bagage pour rentrer les bagages dans le fichier
strcpy(nom2,transit.numerovol);
strcat(nom2,".bin");
for (i=0;i<passage.nbbagage;i++)
{
enregistrement_bagage(passage,nom2,i);
}
//à partir d'ici, on va donner le numéro de sa place à l'utilisateur. Il a deux choix devant lui: choisir
sa
place ou prendre la premiere place disponible.
printf("d%csirez-vous choisir votre place : si oui tapez 1 si non tapez 0\n",130);
```

```
choix=0;
scanf("%d",&choix);
ptr2=fopen(nom2,"r+b");
if (choix == 1)
{
//premier choix : l'utilisateur choisit sa place.
//Ici, on va d'abbord demander à l'utilisateur la place qu'il désire avoir.
printf("vous pouvez choisir une place entre 0 et %d \n",transit.nbplace-1);
scanf("%d",&passage.numplace);
//ensuite, à l'aide du fichier "numerodevol".bin on va rechercher la place sélectionnée par
l'utilisateur et vérifier qu'elle est libre. si ce n'est pas le cas l'utilisateur devra resaisir une place
différente.
fseek(ptr2,sizeof(int),SEEK_SET);
fseek(ptr2,passage.numplace*sizeof(boardpass),SEEK_CUR);
fread(&navig,sizeof(boardpass),1,ptr2);
while(strcmp(navig.nom," ")!=0)
{
printf("cette place est d%cj%c prise, veuillez entrer un nouveau num%cro",130,133,130);
scanf("%d",&passage.numplace);
fseek(ptr2,sizeof(int),SEEK_SET);
fseek(ptr2,passage.numplace*sizeof(boardpass),SEEK_CUR);
fread(&navig,sizeof(boardpass),1,ptr2);
}
printf("Votre place sera donc %d\n",passage.numplace);
}
else
{
//deuxième choix, l'utilisateur ne desire pas choisir sa place
//ici, on se sert du fichier "numerodevol".bin et on cherche la premiere place libre. Cette place sera
affecté à l'utilisateur
fseek(ptr2,sizeof(int),SEEK_SET);
i=0;
fread(&navig,sizeof(boardpass),1,ptr2);
while((strcmp(navig.nom," ")!=0) || (feof(ptr2)!=0))
{
fread(&navig,sizeof(boardpass),1,ptr2);
i++;
}
passage.numplace=i;
printf("votre place sera donc la place num%cro %d\n",130,passage.numplace);
}
//grace au 6 prochaines lignes, on met à jour le nombre de place disponible dans l'avion(inscrit au début
du
fichier "numerodevol".bin)
fseek(ptr2,0,SEEK_SET);
fread(retenu,sizeof(int),1,ptr2);
printf("%d",*retenu);
(*retenu)--;
rewind(ptr2);
fwrite(retenu,sizeof(int),1,ptr2);
//on entre ensuite le boardpass de l'utilisateur dans le fichier à la place qui a préalablement été
définie
fseek(ptr2,passage.numplace*sizeof(boardpass),SEEK_CUR);
fwrite(&passage,sizeof(boardpass),1,ptr2);
fclose(ptr2);
//pour finir, on rentre le boardpas dans le fichier passager.bin où sont enregistré tout les passagers.
ptr3=fopen("passager.bin", "a+b");
fwrite(&passage, sizeof(boardpass), 1, ptr3);
fclose(ptr3);
return passage;
}
/*!
* \fn boardpass achat_billet()
* \brief cette fonction permet de remplir la structure billet qui servira plus tard à remplir la structure
Boardpass
*/
boardpass achat_billet()
{
billet passage;
boardpass passager;
//ici on va rentrer les infos d'un passager dans son billet.
FILE* ptr;
```

```
FILE* ptr2;
vol tab[40];
char nom[20];
int controle,j,i=0;
ptr=fopen("vol.bin","a+b");
system("CLS"); // Clear Screen
printf("Achat Billet\n\n");
//à partir d'ici, on remplit la structure billet
printf("entrer votre pr%cnom\n",130);
scanf("%s",passage.prenom);
printf("entrer votre nom\n");
scanf("%s",passage.nom);
printf("entrer votre date de naissance \n");
printf("entrer le jour : ");
scanf("%d",&passage.naissance[0]);
//les 7 prochaines lignes sont un controle de cohérence pour le jour
while ((passage.naissance[0]>31)||(passage.naissance[0]<0))
{
printf("jour non valide, veuillez entrer un nouveau chiffre");
scanf("%d",&passage.naissance[0]);
}
printf("entrer le mois : ");
scanf("%d",&passage.naissance[1]);
//les lignes 241 à 245 sont un controle de cohérence pour le mois
while ((passage.naissance[1]>12)||(passage.naissance[1]<1))
{
printf("mois non valide, veuillez entrer un nouveau chiffre");
scanf("%d",&passage.naissance[1]);
}
printf("entrer l'ann%ce : ",130);
scanf("%d",&passage.naissance[2]);
printf("entrez votre num%cro de passeport(2 chiffres puis 2 lettres et 5 chiffres)\n",130);
scanf("%s",passage.passeport);
//à partir d'ici, on va afficher les vols disponibles afin que l'utilisateur puisse choisir sa destination
printf("les destinations disponibles sont les suivantes :\n");
fread(&tab[i],sizeof(vol),1,ptr);
j=0;
while (feof(ptr)==0)
{
strcpy(nom,tab[i].numerovol);
strcat(nom,".bin");
ptr2=fopen(nom,"r+b");
fread(&controle,sizeof(int),1,ptr2);
if(controle>0)
{
printf("\t%s\n",tab[i].dest);
j++;
}
i++;
fread(&tab[i],sizeof(vol),1,ptr);
}
if (j==0)
{
printf("aucun vol n'est disponible\n\n");
return passager;
}
else
{
//le passager rentre sa destination
printf("entrer votre destination :\n");
scanf("%s",passage.dest);
// les lignes 262 à 282 constituent un controle au cas où l'utilisateur rentre une destination pour
laquelle aucun vol n'existe.
controle=0;
for (j=0;j<i;j++)
{
if(strcmp(passage.dest,tab[j].dest)==0)
{
controle=1;
}
}
while (controle==0)
```

```
{
printf("destination invalide, veuillez r%cessayer",130);
scanf("%s",passage.dest);
for (j=0;j<i;j++)
{
if(strcmp(passage.dest,tab[j].dest)==0)
{
controle=1;
}
}
}
//Ici, on crée le numero de billet de l'utilisateur et on l'affiche
for(j=0;j<=i;j++)
{
if(strcmp(passage.dest,tab[j].dest)==0)
{
passage.numbillet[0]='N';
passage.numbillet[1]=passage.nom[2];
passage.numbillet[2]=tab[j].numerovol[0];
passage.numbillet[3]=tab[j].numerovol[1];
passage.numbillet[4]=tab[j].numerovol[2];
passage.numbillet[5]=tab[j].numerovol[3];
passage.numbillet[6]=tab[j].numerovol[4];
passage.numbillet[7]=tab[j].numerovol[5];
passage.numbillet[8]=passage.nom[0];
passage.numbillet[9]=passage.prenom[0];
passage.numbillet[10]='\0';
}
}
printf(" votre num%cero de billet est le suivant : %s\n",130,passage.numbillet);
// on finit de completer la structure
printf("entrer votre nationalit%c au masculin, en majuscule et sans caract%cre sp%cciaux\n",130,138,130
);
scanf("%s",passage.nationalite);
fclose(ptr);
fclose(ptr2);
//on apelle la fonction boarpass
passager=creation_boardpass(passage);
return passager;
}
}
/*!
* \fn void frontieres ()
* \brief cette fonction imite le passage de frontières pour un passager et permet de conseiller le
passager
sur le besoin ou non d'un visa.
*/
void frontieres ()
//Ici on détermine si oui ou non le passager a besoin d'un visa selon sa nationalité et sa destination. On
affichera sa destination, sa nationnalité et s'il a besoin d'un visa.
{
FILE* fichier;
boardpass passager;
system("CLS"); // Clear Screen
fichier=fopen("passager.bin","a+b");
//je ne dois pas copier le paragraphe suivant, il me sert juste à remplir mon fichier (normalemnt ce
fichier sera remplit par le programme boardpass)
fread(&passager,sizeof(boardpass),1,fichier);
while (feof(fichier)==0)
{
printf("Le passager %s %s a pour nationalit%c : %s.\nSa destination est %s.\n", passager.prenom,
passager.nom, 130, passager.nationalite, passager.dest);
if (strcmp(passager.nationalite, "AMERICAIN")==0)
{
if(strcmp(passager.dest, "USA")==0 ||strcmp(passager.dest, "ETATS_UNIS")==0 || strcmp(passager.dest,
"US")==0 || strcmp(passager.dest, "EU")==0) printf("Le passager n'a pas besoin de visa\n\n");
else printf("Le passager a besoin d'un visa\n\n");
}
//Le cas où le passager est européen (francais, anglais, espagnol, allemand)
if (strcmp(passager.nationalite, "FRANCAIS")==0 || strcmp(passager.nationalite, "ANGLAIS")==0 || strcmp(
passager.nationalite, "ALLEMAND")==0 || strcmp(passager.nationalite, "ESPAGNOL")==0)
{
```

```
if(strcmp(passager.dest, "FRANCE")==0 || strcmp(passager.dest, "ANGLETERRE")==0 || strcmp(passager.
dest, "ALLEMAGNE")==0 || strcmp(passager.dest, "ESPAGNE")==0) printf("Le passager n'a pas besoin de
visa\n\n");
else printf("Le passager a besoin d'un visa\n\n");
}
if (strcmp(passager.nationalite, "CHINOIS")==0)
{
if(strcmp(passager.dest, "CHINE")==0) printf("Le passager n'a pas besoin de visa\n\n");
else printf("Le passager a besoin d'un visa\n\n");
}
fread(&passager, sizeof(boardpass),1,fichier);
}
fclose(fichier);
}
/*!
* \fn void securite()
* \brief cette fonction imite le passage de la sécurité pour un passager. Cela inclut le controle des
affaires
et notament le passage du sac au travers le scanner
*/
void securite()
{
int nb_affaire,i;
char affaire[100];
system("CLS");
printf("Les passagers doivent traverser la s%ccurit%c, leurs bagages %c main sont pass%cs au scanner.\nOn
se
propose ici de simuler ce passage...\nPour cela, vous jouerez le role du scanner. Autrement dit c'est vous
qui
indiquerez le contenu du bagage %c main.\nTout d'abord, saisir le nombre d'objets contenus dans le bagage
%c
main : ",130,130,133,130,133,133);
scanf("%d", &nb_affaire);
printf("\nVous allez saisir le nom des affaires contenues dans le bagage %c main.\n",133);
printf("Voici quelques r%cgles a respecter...\nSi l'objet est compos%c de plusieurs mots, remplacer les
espaces par des underscores et %ccrire en minuscule. Certains objets interdits en cabine ont %ct%c
rassembl%cs
en grande cat%cgorie. V%crifiez bien que le produit saisi n'en fait pas partie. Si c'est le cas respectez
la
d%cnomination donn%ce dans les informations.",138,130,130,130,130,130,130,130,130,130);
for (i=0; i<nb_affaire ; i++){
printf("\n\nPour information : \nSi le passager transporte une arme a feu saisir simplement :
arme_a_feu\nSi
le passager transporte une arme %clectrique saisir simplement : arme_electrique\nSi le passager transporte
une
arme tranchante saisir simplement : arme_tranchante\nSi le passager transporte un objet tranchant saisir
simplement : objet_tranchant\nSi le passager transporte un instrument contondant saisir simplement
:objet_contondant\nSi le passager transporte un article pyrotechnique saisir simplement :
article_pyrotechnique\nSi le passager transporte un engin de deplacement avec batterie saisir simplement :
engin_de_deplacement_avec_batterie\nSi le passager transporte une boisson alcoolis%ce saisir simplement :
alcool\nSi le passager transporte un produit m%cnager saisir simplement : produit_menager\nSi le passager
transporte un article de bricolage saisir simplement : produit_pour_bricolage\n\n",130,130,130);
printf("\nQuel est le nom de l'objet %d ? ", i+1);
scanf("%s", affaire);
if (strcmp(affaire, "aerosol")==0 || strcmp(affaire, "oxygene_liquide")==0 || strcmp(affaire,
"thermometre_a_mercure")==0 || strcmp(affaire, "arme_a_feu")==0 || strcmp(affaire, "arme_electrique")==0
||
strcmp(affaire, "arme_tranchante")==0 || strcmp(affaire, "objet_tranchant")==0 || strcmp(affaire,
"objet_contondant")==0 || strcmp(affaire, "article_pyrotechnique")==0 || strcmp(affaire,
"engin_de_deplacement_avec_batterie")==0 || strcmp(affaire, "alcool")==0 || strcmp(affaire,
"produit_menager")==
0 || strcmp(affaire, "produit_pour_bricolage")==0){
printf("Il faudra retirer ce produit.\n");
}
else printf("Cet objet est autoris%c.\n",130);
}
}
/*!
* \fn void decollage(char* nom, boardpass *tabpass, bagage* tabbag, int nb_passager_emb,int choix )
* \brief cette fonction permet de simuler le decollage d'un vol. Pour cela on verifie que l'embarquement
s'est
bien déroulé et que tout les passagers sont transférées, puis on supprime les fichiers inutiles.
```

```
*/
void decollage(char* nom, boardpass *tabpass, bagage* tabbag, int nb_passager_emb,int nb_bag_emb, int
choix )
//ici on fait simplement quelques printf poyur indiquer si l'avion peut décoller.
{
int nb_place,nb_avion_emb,nb_bag,pret,i;
FILE *efichier;
FILE *fichier;
FILE *fnew;
char nom_emb[20];
char nom_file[20];
char nomvol[20];
vol navig;
boardpass pass_emb;
boardpass pass_navig;
bagage b_circul;
system("CLS");
//Selon la décision de l'utilisateur dans la fonction embarquement, on distingue deux cas :
//Soit l'utilisateur veut décoller l'avion tout de suite apres l'embarquement donc choix=1
//Soit l'utilisateur ne veut pas décoller l'avion tout de suite apres l'embarquement. Quand il a change
d'avis, il accede a la fonction decollage via le menu. Dans ce cas choix=0
//Si l'utilisateur n'a pas fait d'embarquement préalablement, on entre dans le cas choix=0
if (choix==0){
//On affiche les vol qui ont été embarqués préalablement
fichier=fopen("vol.bin", "rb");
nb_avion_emb=0;
printf("Voici les avions embarqu%cs : \n",130);
fread(&navig,sizeof(vol),1,fichier);
while (feof(fichier)==0)
{
strcpy(nom_emb,"E");
strcpy(nom_file,navig.numerovol);
strcat(nom_file,".bin");
strcat(nom_emb, nom_file);
efichier=fopen(nom_emb, "r+b");
if (efichier!=NULL){
printf("%s\n",nom_file);
nb_avion_emb++;
}
fclose(efichier);
fread(&navig,sizeof(vol),1,fichier);
}
//ici, aucun avion n a été embarqué, on indique à l'utilisateur quaucun vol ne peut décoller
if (nb_avion_emb==0)
{
printf("Aucun avion n'est pret a d%ccoller. Embarquez pr%calablement un avion\n\n",130,130);
pret=0;
}
//Dans ce cas, un ou plusieurs avions ont été embarqué, on doit donc déterminer quelle avion l'utilisateur
veut il faire décoller
else
{ pret=1;
printf("Quelle avion voulez-vous faire d%ccoller ?\n",130);
scanf("%s",nom_file);
strcpy(nomvol,nom_file);
//Maintenant qu'on connait le numero de vol, on détermine le nombre de place total dans l'avion qui
s'apprete a decoller
rewind(fichier);
fread(&navig,sizeof(vol),1,fichier);
while (strcmp(navig.numerovol,nom_file)!=0)
{
fread(&navig,sizeof(vol),1,fichier);
}
nb_place=navig.nbplace;
//Ici, on cherche a déterminer le nombre de personne qui ont embarqué
strcat(nom_file,".bin");
strcpy(nom_emb,"E");
strcat(nom_emb, nom_file);
efichier=fopen(nom_emb,"rb");
fread(&pass_emb, sizeof(boardpass),1,efichier);
while (feof(efichier)==0)
{
```

```
nb_passager_emb++;
fread(&pass_emb, sizeof(boardpass),1,efichier);
}
if (nb_passager_emb==nb_place) printf("Tous les passagers ont embarqu%c.\n",130);
fclose(efichier);
//Maintenant, il vaut vérifier que tous les bagages des passagers sont stockés dans l'avion
//Ici on détermine le nombre de bagage que doit contenir l'avion avant de décoller
nb_bag=0;
efichier=fopen(nom_file, "rb");
fseek(efichier, sizeof(int), SEEK_SET);
fseek(efichier, nb_place*sizeof(boardpass),SEEK_CUR);
while (feof(efichier)==0)
{
fread(&b_circul,sizeof(bagage),1,efichier);
nb_bag++;
}
fclose(efichier);
//Ici on détermine le nombre de bagage embarque
strcpy(nom_emb,"BE");
strcat(nom_emb, nom_file);
efichier=fopen(nom_emb,"rb");
fread(&b_circul, sizeof(bagage),1,efichier);
while (feof(efichier)==0)
{
nb_bag_emb++;
fread(&b_circul,sizeof(bagage),1,efichier);
}
fclose(efichier);
if (nb_bag==nb_bag_emb)printf("Tous les bagages ont %ct%c embarqu%c.\n",130,130,130);
fclose(fichier);
}
}
//Dans cette condition, l'utilisateur a choist de faire décoller son avion tout de suite après
l'embarquement, nous récuppérons donc de la fonction embarquement un certain nombre d'information
else
{
//Vérifions que tous les personnes qui ont achete un ticket ont embarqué
pret=1;
fichier=fopen("vol.bin", "rb");
fread(&navig,sizeof(vol),1,fichier);
while (strcmp(nom,navig.numerovol)!=0)fread(&navig,sizeof(vol),1,fichier);
if (navig.nbplace==nb_passager_emb)printf("Tous les passagers ont embarqu%cs.\n",130);
printf("%d",nb_passager_emb);
fclose(fichier);
//Maintenant, il vaut vérifier que tous les bagages des passagers sont stockés dans l'avion
nb_bag=0;
strcpy(nom_file,nom);
strcat(nom_file,".bin");
fichier=fopen(nom_file, "rb");
fseek(fichier, sizeof(int), SEEK_SET);
fseek(fichier, navig.nbplace*sizeof(boardpass),SEEK_CUR);
while (feof(fichier)==0)
{
nb_bag++;
fread(&b_circul,sizeof(bagage),1,fichier);
}
fclose(fichier);
nb_bag--;
if (nb_bag==nb_bag_emb)printf("Tous les bagages ont %ct%c embarqu%c.\n",130,130,130);
strcpy(nomvol,nom);
}
//Toutes les conditions pour décoller sont respecter, on peut procéder a la simulation du décollage
if (pret==1){
printf("\n\nD%ccollage en cour...\n",130);
//Nous devons supprimer des informations relatifs a l'avion qui est sur le point de decoller
//La 1ere etape consiste a supprimer le vol du fichier vol.bin (qui rassemble toutes nos structure vol)
fnew=fopen("newvol.bin","a+b");
fichier=fopen("vol.bin","r+b");
fread(&navig,sizeof(vol), 1, fichier);
while (feof(fichier)==0)
{
if (strcmp(navig.numerovol,nomvol)!=0) fwrite(&navig, sizeof(vol),1,fnew);
```

```
fread(&navig,sizeof(vol), 1, fichier);
}
fclose(fnew);
fclose(fichier);
remove("vol.bin");
rename("newvol.bin","vol.bin");
//La 2eme partie consiste a supprimer les passager du vol qui sont contenus daans le fichier passager.bin
(qui rassemble toutes nos structures boardpass)
if (choix==1)
{
for (i=0;i<nb_passager_emb;i++)
{
fnew=fopen("newpassager.bin","a+b");
fichier=fopen("passager.bin","r+b");
fread(&pass_navig,sizeof(boardpass),1,fichier);
while (feof(fichier)==0)
{
if(strcmp(pass_navig.passeport,tabpass[i].passeport)!=0)fwrite(&pass_navig,sizeof(boardpass),1,
fnew);
fread(&pass_navig,sizeof(boardpass),1,fichier);
}
fclose(fnew);
fclose(fichier);
remove("passager.bin");
rename("newpassager.bin","passager.bin");
}
}
else
{
strcpy(nom_emb,"E");
strcat(nom_emb, nom_file);
efichier=fopen(nom_emb, "r+b");
for (i=0;i<nb_passager_emb;i++)
{
fnew=fopen("newpassager.bin","a+b");
fichier=fopen("passager.bin","r+b");
fread(&pass_emb,sizeof(boardpass),1,efichier);
fread(&pass_navig,sizeof(boardpass),1,fichier);
while (feof(fichier)==0)
{
if(strcmp(pass_navig.passeport,pass_emb.passeport)!=0)fwrite(&pass_navig,sizeof(boardpass),1,
fnew);
fread(&pass_navig,sizeof(boardpass),1,fichier);
}
fclose(fnew);
fclose(fichier);
remove("passager.bin");
rename("newpassager.bin","passager.bin");
}
fclose(efichier);
}
//Pour simuler le decollage nous allons supprimer tous les fichiers relatifs au vol en question
remove(nom_file);
if (choix==0)
{
remove(nom_emb);
strcpy(nom_emb,"BE");
strcat(nom_emb, nom_file);
remove(nom_emb);
}
printf("L'avion vient de d%ccoller.\n\n",130);
}
}
/*!
* \fn boardpass* embarquement
* \brief cette fonction permet de simuler l'embarquement. Pour cela on transfère les données stockées dans
les
fichiers dans des tableaux dynamiques.
*/
boardpass* embarquement()
{
//ici on remet tout les passagers dans un meme tableau à partir du fichier "numerodevol".bin
```

```
FILE* ptr;
FILE* ptr2;
FILE* ptremb;
boardpass* passage, passager;
bagage *tab;
bagage transit;
char nom [20];
char nom2[20];
char nomemb[20];
int retenu,i,j,choix;
vol navig;
system("CLS");
// en premier lieu on va afficher les vols qui ne possèdent plus de places à vendre et qui sont, de ce
fait,
pret pour l'embarquement
ptr=fopen("vol.bin","r+b");
fread(&navig,sizeof(vol),1,ptr);
printf("Les vols disponibles pour l'embarquement sont : \n");
i=0;
while (feof(ptr)==0)
{
strcpy(nom,navig.numerovol);
strcat(nom,".bin");
ptr2=fopen(nom,"r+b");
fread(&retenu,sizeof(int),1,ptr2);
if (retenu==0)
{
printf("\n\t %s",nom);
i++;
}
fclose(ptr2);
fread(&navig,sizeof(vol),1,ptr);
}
//on demande à l'utilisateur le vol qu'il veut faire embarquer. Deux cas se présentent : soit il n'y a pas
de vols à faire embarquer soit il y en a
if (i==0)
{
//cas1 : il n'y a pas de vol à faire embarquer, on renvoie donc l'utilisateur au menu
printf("Aucun vol ne peut embarquer car aucun vol n'est complet.\n");
return 0;
}
else
{
//cas 2 : il y a des vols à faire embarquer
printf("\nEntrer le numero du vol a faire embarquer (ne pas saisir l'extension .bin)\n");
scanf("%s",nom);
strcpy(nom2,nom);
strcat(nom,".bin");
ptr2=fopen(nom,"r+b");
//Ici on recherche le nombre de place dans l'avion pour créer le tableau dynamique
fseek(ptr2,sizeof(int),SEEK_SET);
strcpy(navig.numerovol,"000000");
while(strcmp(nom2,navig.numerovol)==0)
{
fread(&navig,sizeof(vol),1,ptr);
}
passage=(boardpass*)malloc(navig.nbplace*sizeof(boardpass));
//à partir d'ici, on remplit le tableau avec d'abbord les passagers priority dans la première boucle for
puis avec les passagers normaux dans la deuxieme boucle for
j=0;
fseek(ptr2,sizeof(int),SEEK_SET);
for(i=0;i<navig.nbplace;i++)
{
fread(&passager,sizeof(boardpass),1,ptr2);
if(passager.privilege==1)
{
passage[j]=passager;
j++;
}
}
fseek(ptr2,sizeof(int),SEEK_SET);
for(i=0;i<navig.nbplace;i++)
```

```
{
fread(&passager,sizeof(boardpass),1,ptr2);
if(passager.privilege==0)
{
passage[j]=passager;
j++;
}
}
printf("\nVoici la liste des passagers embarques : \n");
for(i=0;i<navig.nbplace;i++)
{
printf("nom : %s\n",passage[i].nom);
printf("prenom : %s\n\n",passage[i].prenom);
}
printf("Embarquement des bagages\n");
fseek(ptr2,sizeof(int),SEEK_SET);
fseek(ptr2,navig.nbplace*sizeof(boardpass),SEEK_CUR);
i=0;
while(feof(ptr2)==0)
{
fread(&transit,sizeof(bagage),1,ptr2);
i++;
}
tab=(bagage*)malloc((i+1)*sizeof(bagage));
fseek(ptr2,sizeof(int),SEEK_SET);
fseek(ptr2,navig.nbplace*sizeof(boardpass),SEEK_CUR);
for(j=0;j<i;j++)
{
fread(&tab[i],sizeof(bagage),1,ptr2);
}
j=i;
fclose(ptr);
fclose(ptr2);
printf("L'embarquement de l'avion est termin%ce. Voulez-vous proc%cdez au d%ccolage ? entrez 1 si oui et
0 si non\n",130,130,130);
scanf("%d",&choix);
if (choix==1)
{
decollage(nom2, passage,tab,navig.nbplace,j-1, 1);
}
else
{
strcpy(nomemb,"E");
strcat(nomemb,nom);
ptremb=fopen(nomemb,"a+b");
for(i=0;i<navig.nbplace;i++)
{
fwrite(&passage[i],sizeof(boardpass),1,ptremb);
}
fclose(ptremb);
strcpy(nomemb,"BE");
strcat(nomemb,nom);
ptremb=fopen(nomemb,"a+b");
for(i=0;i<j;i++)
{
fwrite(&tab[i],sizeof(bagage),1,ptremb);
}
fclose(ptremb);
}
return passage;
}
}
/*!
* \fn int main()
* \brief fonction principale du programme qui servira de menu pour naviquer entre les différentes
fonctions.
*/
int main()
{
char choix;
choix='0';
while (choix!='Q')
```

```
{
printf("Gestion des Vols\n\n\tC: cr%cation de vol,\n\tA: achat billet / boardpass / enregistrement
bagage, \n\tF: passage de fronti%cres, \n\tS: s%ccurit%c, \n\tE: embarquement, \n\tD: d%ccollage, \n\tQ:
quitter\nQue souhaitez vous faire ?",138,130,130,130,130);
scanf("%s",&choix);
choix=toupper(choix);
switch (choix)
{
case 'A' : achat_billet(); break;
case 'F' : frontieres(); break;
case 'S' : securite(); break;
case 'D' : decollage(NULL, NULL, NULL,0,0,0); break;
case 'E' : embarquement(); break;
case 'C' : creer_vol(); break;
case 'Q' : printf("%c bient%ct\n",133,147) ;break;
default : printf(" commande inexistante\n");
}
}
return 0;
}
```