Projet du cours de principes algorithmiques et programmation :

On vous demande d'écrire un programme permettant de répondre à la situation suivante :

Soient 2 banques belges A et B ayant l'intention de fusionner. La banque A souhaite avertir, par l'envoi d'une lettre personnalisée, les clients disposant d'un numéro de compte à vue dans chacune des 2 banques. Dans cette lettre, il est signifié au client que celui-ci aura la possibilité, à l'issue de la fusion, soit de conserver les 2 numéros de compte à vue, soit de fermer, au choix, un des 2 numéros de compte.

La banque A dispose d'un fichier client contenant les informations concernant les clients ayant un numéro de compte à vue dans la banque A. Ce fichier client est constitué d'un numéro de client, d'un numéro de compte bancaire, d'un nom de client, d'un prénom, d'une date de naissance et d'un numéro de registre national. La banque A a également à sa disposition un fichier client trié par ordre croissant des numéros de client ainsi qu'un fichier client trié par ordre alphabétique des noms de client.

Le numéro de client est un nombre entier tandis que le nom et le prénom des clients sont constitués de 30 caractères au maximum.

Par souci de simplification, les noms des clients seront constitués uniquement de caractères choisis parmi les 27 caractères suivants : l'espace blanc (code ascii 32) et les 26 lettres majuscules non accentuées de l'alphabet (code ascii compris entre 65 et 90). Un nom de client ne peut pas commencer par un espace blanc.

Par souci de simplification, les prénoms des clients seront constitués uniquement de caractères choisis parmi les 28 caractères suivants : l'espace blanc (code ascii 32), le tirait (code ascii 45) et les 26 lettres majuscules non accentuées de l'alphabet (code ascii compris entre 65 et 90). Un prénom de client ne peut pas commencer par un espace blanc ni par un tirait.

Les dates de naissance sont codées sous la forme jj/mm/aaaa (exemple : 05/04/1971). Une date de naissance de client ne sera valide que si elle est postérieure au 31 décembre 1904 et antérieure au 1er janvier 1997. En outre, des dates comme le 29/02/1987 ou le 31/11/1970 ne sont évidemment pas valides.

Le numéro de compte bancaire est constitué de 19 caractères : la lettre B puis la lettre E suivie de 2 chiffres, un espace, puis 4 chiffres suivis d'un espace, 4 chiffres, un espace et à nouveau 4 chiffres
(exemple : BE86 0643 4587 4856).

Le numéro de registre national est constitué de 13 caractères : 6 chiffres suivis d'un tirait, lui-même suivi de 3 chiffres, d'un tirait et de 2 chiffres (exemple : 700914-158-78).

Dans un premier temps, vous créerez les fichiers client de la banque A. Pour ce faire, vous

- utiliserez une structure clientA définie comme suit :

```
struct clientA
{
        int numero ;
        char nom [31];
        char prenom [31];
        char datenaiss [11] ;
        char num_compte [20] ;
        char num_reg_nat [14] ;
};
```

- créerez une fonction nommée « verifnom » dont le prototype est le suivant :

  ```
  int  verifnom(char* chaine) ;
  ```

  Cette fonction retourne la valeur 0 si la chaîne de caractères ne correspond pas à un nom de client valide et retourne la valeur 1 dans le cas contraire ;

- créerez une fonction nommée « verifprenom » dont le prototype est le suivant :

  ```
  int  verifprenom(char* chaine) ;
  ```

  Cette fonction retourne la valeur 0 si la chaîne de caractères ne correspond pas à un prénom de client valide et retourne la valeur 1 dans le cas contraire ;

- créerez une fonction nommée « verifdatenaiss » dont le prototype est le suivant :

  int verifdatenaiss (char* chaine) ;

  Cette fonction retourne la valeur 0 si la chaîne de caractères ne correspond pas à une date de naissance valide et retourne la valeur 1 dans le cas contraire ;

- créerez une fonction nommée « verifnumcompte » dont le prototype est le suivant :

  int verifnumcompte (char* chaine) ;

  Cette fonction retourne la valeur 0 si la chaîne de caractères ne correspond pas à un numéro de compte bancaire valide et retourne la valeur 1 dans le cas contraire ;

- créerez une fonction nommée « verifnumregnat » dont le prototype est le suivant :

  int verifnumregnat (char* chaine) ;

  Cette fonction retourne la valeur 0 si la chaîne de caractères ne correspond pas à un numéro de registre national valide et retourne la valeur 1 dans le cas contraire ;

- créerez une fonction nommée « compatibildatenaissregnat» dont le prototype est le suivant :

  int compatibildatenaissregnat (char* dateofbirth, char* numregnat) ;

  Cette fonction retourne la valeur 1 si la date de naissance est compatible avec le numéro de registre national (Les deux premiers chiffres du numéro de registre national correspondent à l'année de naissance ; les deux suivants au mois de la naissance tandis que le $5^{ème}$ et $6^{ème}$ chiffre correspondent au jour de la naissance) et retourne la valeur 0 dans le cas contraire (Ainsi, par exemple, le numéro de registre national 710712-125-24 est compatible avec la date de naissance 12/07/1971 tandis que le numéro de registre national 710712-125-24 est incompatible avec la date de naissance 07/12/1971) ;

- créerez une fonction nommée « encodenouvclientA» dont le prototype est le suivant :

  struct clientA encodenouvclientA (void) ;

  Cette fonction permet à l'utilisateur d'encoder les informations concernant un nouveau client de la banque A. La fonction vérifie la validité des informations et invite l'utilisateur à se corriger si certaines informations sont erronées. Cette fonction fera notamment appel aux fonctions verifnom, verifprenom, verifdatenaiss , verifnumcompte , verifnumregnat et compatibildatenaissregnat évoquées précédemment. Les « saisies vides » (lorsque l'utilisateur appuie sur la touche ENTER sans avoir préalablement saisi de caractère) sont interdites pour chaque information concernant un client de la banque A (autrement dit pas de « saisie vide » pour le numéro, le nom, le prénom, la date de naissance, le numéro de compte bancaire et le numéro de registre national). Les informations valides concernant le nouveau client de la banque A seront contenues dans la structure clientA retournée par la fonction ;

- créerez une fonction nommée « stringcopy» dont le prototype est le suivant :

  void stringcopy (char* source, char* destination) ;

  Cette fonction permet de copier les caractères se trouvant dans la chaîne de caractères source (y compris la marque de fin de chaîne) et de les placer dans la chaîne de caractères destination ;

- créerez une fonction nommée « structclientAcopy» dont le prototype est le suivant :

  void structclientAcopy (struct clientA source, struct clientA* destination) ;

  Cette fonction permet de copier les informations se trouvant dans la structure clientA source et de les placer dans la structure clientA dont destination contient l'adresse. Vous ferez appel notamment à la fonction stringcopy évoquée précédemment;

- créerez trois tableaux nommés respectivement sourceA, trinumA, trinomA de M (où M est une constante correspondant au nombre maximum de clients dans la banque A) structures clientA ;
- demanderez à l'utilisateur, au cours de l'exécution du programme, de saisir le nombre n de clients de la banque A dont il souhaite encoder les informations. Ce nombre n devra être inférieur ou égal à M. Si tel n'est pas le cas, un message avertira l'utilisateur et une nouvelle saisie sera effectuée tant que n > M ;
- remplirez le tableau sourceA de structures clientA en demandant à l'utilisateur de saisir les informations concernant les clients de la banque A. Vous utiliserez pour ce faire les fonctions encodenouvclientA et structclientAcopy ;
- remplirez le tableau trinumA en copiant dans un premier temps toutes les informations contenues dans le tableau sourceA (vous utiliserez pour ce faire la fonction structclientAcopy) et en effectuant dans un deuxième temps un tri croissant par numéro de client ;
- remplirez le tableau trinomA en copiant dans un premier temps toutes les informations contenues dans le tableau sourceA (vous utiliserez pour ce faire la fonction structclientAcopy) et en effectuant dans un deuxième temps un tri par ordre alphabétique des noms de client. Pour réaliser ce tri, vous créerez et utiliserez une fonction que vous nommerez « stringcomp » permettant de signaler si une chaîne de caractères donnée A précède (ou succède), dans l'ordre lexicographique, une autre chaîne de caractères B également donnée (la fonction renvoie la valeur 0 si les chaînes A et B sont identiques ; la fonction renvoie 1 si A précède B dans l'ordre lexicographique et enfin, la fonction renvoie –1 si A succède B dans l'ordre lexicographique. Ainsi, si A est « coucou » et B est « cou », la fonction renvoie –1 ; si A est « argile » et B est « charbon », la fonction renvoie 1 et enfin, si A est « bien » et B est « bien », la fonction renvoie 0). Cette fonction aura comme prototype : ;

  int stringcomp (char* A, char* B) ;

- afficherez les informations concernant les clients de la banque A des trois tableaux évoqués ci-dessus ;
- créerez 3 fichiers client contenant respectivement les informations des clients de la banque A contenues dans le tableau sourceA, le tableau trinumA et le tableau trinomA.

La banque A dispose également d'un fichier contenant le nom, le prénom et la date de naissance des clients de la banque B possédant un compte à vue dans cette banque B. Les contraintes sur les noms, les prénoms et les dates de naissance des clients de la banque B sont les mêmes que celles sur les noms, prénoms et dates de naissance des clients de la banque A.
Dans un deuxième temps, vous créerez ainsi le fichier client de la banque B. Pour ce faire, vous

- utiliserez une structure clientB définie comme suit :

```
struct clientB
{
        char nom [31];
        char prenom [31];
        char datenaiss [11] ;
};
```

- créerez une fonction nommée « encodenouvclientB» dont le prototype est le suivant :

  struct clientB encodenouvclientB (void) ;

  Cette fonction permet à l'utilisateur d'encoder les informations concernant un nouveau client de la banque B. La fonction vérifie la validité des informations et invite l'utilisateur à se corriger si certaines informations sont erronées. Cette fonction fera notamment appel aux fonctions verifnom, verifprenom, et verifdatenaiss évoquées précédemment. Les « saisies vides » (lorsque l'utilisateur appuie sur la touche ENTER sans avoir préalablement saisi de caractère) sont interdites pour chaque information concernant un client de la banque B (autrement dit pas de « saisie vide » pour le nom, le prénom et la date de naissance). Les informations valides concernant le nouveau client de la banque B seront contenues dans la structure clientB retournée par la fonction ;

- créerez une fonction nommée « structclientBcopy» dont le prototype est le suivant :

  void structclientBcopy (struct clientB source, struct clientB* destination) ;

  Cette fonction permet de copier les informations se trouvant dans la structure clientB source et de les placer dans la structure clientB dont destination contient l'adresse. Vous ferez appel notamment à la fonction stringcopy évoquée précédemment;

- créerez un tableau nommé sourceB de P (où P est une constante correspondant au nombre maximum de clients dans la banque B) structures clientB ;
- demanderez à l'utilisateur, au cours de l'exécution du programme, de saisir le nombre q de clients de la banque B dont il souhaite encoder les informations. Ce nombre q devra être inférieur ou égal à P. Si tel n'est pas le cas, un message avertira l'utilisateur et une nouvelle saisie sera effectuée tant que q > P ;
- remplirez le tableau sourceB de structures clientB en demandant à l'utilisateur de saisir les informations concernant les clients de la banque B. Vous utiliserez pour ce faire les fonctions encodenouvclientB et structclientBcopy ;
- afficherez les informations concernant les clients de la banque B contenues dans le tableau sourceB.
- créerez un fichier client contenant les informations des clients de la banque B contenues dans le tableau sourceB.

La banque A désire connaître les personnes qui ont un numéro de compte dans chacune des 2 banques.
Elle dispose pour cela de ses 3 fichiers client évoqués précédemment ainsi que du fichier client contenant le nom, le prénom et la date de naissance des clients de la banque B. Elle veut donc créer un fichier contenant les informations (nom, prénom et date de naissance) des clients possédant un numéro de compte dans chacune des 2 banques.
Deux personnes seront considérées comme identiques si elles ont le même nom, le même prénom et la même date de naissance. Pour créer ce fichier, vous

- créerez un tableau de structures clientB nommé « clients_communs » dans lequel vous stockerez les informations des clients ayant un numéro de compte dans chacune des 2 banques ;
- afficherez les informations des clients présents dans le tableau « clients_communs ».
- créerez un fichier contenant les informations des clients contenues dans le tableau « clients_communs ».

En outre, votre programme doit être capable de lire les informations contenues dans les différents fichiers client et de les afficher à l'écran.

Le programme doit fonctionner avec **le compilateur de l'école**. Tout programme présentant des erreurs lors de la compilation sera sanctionné par une cote nulle.