

```

/*
Exercice sur la gestion des comptes du bar :
pouvoir stocker des fiches contenant un nom, un prenom et la dette au bar

Auteur : Driancourt Thomas alias Velour
Compilateur : Borland
Date : Mai 2005
*/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>

typedef struct
{
    char id[5]; // id représente le numéro de la place occupé
réellement par la fiche dans le fichier fiche.txt
    char name[30];
    char firstname[30];
    char money[20];
}FICHE;

/* Remarquons que la structure ne contient que des chaines de caractère et non
pas un flottant pour la dette, cela pour éviter
de retrouver de manière spontanée et innattendue un caractère EOF quelque part
dans le fichier, c'est aussi la raison pour laquelle
j'ai choisis de créer une fonction pour initialiser les chaines de caractère
avant de les écrire */

// Fonction ajouter une fiche ( au cas ou echo ne s'en doutait pas :) )
void add(FICHE record);

// Fonction initialiser une fiche
void init_fiche(FICHE * a);

// Fonction afficher toutes les fiches
void print_all();

//Fonction afficher une fiche particulière
void print(int a);

//Fonction supprimer une fiche
int delet(int a);

//Fonction supprimer toutes les fiches
void delet_all();

// Fonction de purge
void flush_fich();

FILE * file;
FILE * index; // Les 5 premiers caracteres qui representent le
last_id c'est-a-dire l'id attribué à la dernière fiche
int last_id;

int main()

```

```

{
char op='1', convert[5], tab[30];
FICHE a;
int fiche;

system("CLS");
fflush(stdin); // purge le buffer d'entre standard

/*****/
//ouverture du fichier en mode add, si le fichier n'existe pas il est crée.

file=fopen("C:\\fiche.txt","a");
fclose(file);
// maintenant on peut ouvrir en r+, on est sure que le fichier existe.
file=fopen("C:\\fiche.txt","r+");
/*****/

/*****/
// meme philosophie pour le fichier d'index
index=fopen("C:\\index.txt","a");
fclose(index);
index=fopen("C:\\index.txt","r+");
/*****/

if(file == NULL && index == NULL )
{
printf("Erreur d'ouverture de fichier veuillez sacrifier un poulet et recommencer\n");
exit(1);
}

fread(&convert,sizeof(char),5,index);
last_id=atoi(convert);

while(op != '0')
{
system("CLS");
printf("\t *****\n\t * MENU *\n\t *****\n\n");
printf("entrez 0 pour quitter\n");
printf("entrez 1 pour ajouter une fiche\n");
printf("entrez 2 pour afficher toutes les fiches actives\n");
printf("entrez 3 pour afficher une fiche particuliere\n");
printf("entrez 4 pour supprimer une fiche particuliere\n");
printf("entrez 5 pour supprimer Toutes les fiches\n");
printf("entrez 6 pour purger la base de données\n");
op=getchar();

switch(op)
{
case '1': printf("Entrez le nom\n");
scanf("%s",a.name);

printf("Entrez le prenom\n");
scanf("%s",a.firstname);

printf("Entrez le montant\n");
scanf("%s",a.money);

```

```

        add(a);
        break;

    case '2':    print_all();
                break;

    case '3':    printf("entrez le numéro de la fiche que vous voulez
afficher\n");
                scanf("%d",&fiche);
                print(fiche);
                break;

    case '4':    printf("entrez le numéro de la fiche que vous voulez
supprimer\n");
                scanf("%d",&fiche);
                delet(fiche);
                break;

    case '5':    delet_all();
                break;

    case '6':    flush_fich();
                break;
    }
}
fclose(file);
fclose(index);
system("CLS");
printf("Fin de programme\nAppuyer sur une touche");
getch();
return 0;
}

```

```

// Fonction ajouter une fiche -----
void add(FICHE record)
{
    FICHE a;
    char tab[5];
    int b;

    init_fiche(&a);
    strcpy(a.name, record.name);
    strcpy(a.firstname, record.firstname);
    strcpy(a.money, record.money);

    last_id++;
    itoa(last_id, a.id, 10);           // conversion du last_id en une chaine de
caractère en base 10

    // écriture de la fiche dans le fichier file
    fseek(file, 0L, SEEK_END);
    fwrite(&a, sizeof(FICHE), 1, file);

    // écriture de l'id et du last_id dans le fichier index
    fseek(index, 0, SEEK_SET);
    fwrite(a.id, sizeof(char), 5, index);
}

```

```

fseek(index,5,SEEK_SET);

while(!feof(index))
{
    b=fread(tab, sizeof(char),5,index);
    if (b < 5)
    {
        fwrite(a.id, sizeof(char),5,index);
        break;
    }
}

printf("Appuyer sur une touche");
getch();
}
// -----

// Fonction initialiser un fiche -----
void init_fiche(FICHE * a)
{
    int i;

    for(i=0; i < 30; i++)
    {
        a->name[i]='\0';
        a->firstname[i]='\0';
        if(i<20)
        {
            a->money[i]='\0';
            if(i<5) a->id[i]='\0';
        }
    }
}
// -----

// Fonction print_all -----

/*
On parcourt l'index pour récupérer les fiches actives, l'index ne contient que
des
fiches actives.
*/
void print_all()
{
    FICHE a;
    char convert[5];
    int b=1, _offset;

    fseek(index,5,SEEK_SET); // on passe au dessus des 5
premières caracteres qui representent le last_id
    while(!feof(index))
    {
        b=fread(convert, sizeof(char),5,index);

```

```

        if (b == 5)
        {
            _offset=atoi(convert);
            printf("offset : %d\n",_offset);
            fseek(file, (_offset-1)*sizeof(FICHE),SEEK_SET);
// -1 car le last_id commence a 1, la premiere fiche correspond a un deplacement de 0.
            fread(&a, sizeof(FICHE),1,file);
            printf("\tnom : %s\n\tprenom : %s\n\tmontant : %s\n\n", a.name,
a.firstname, a.money);
        }
        printf("Plus de fiche a afficher, appuyez sur une touche.");
        getch();
    }
// -----

// Fonction afficher une fiche particulière -----
/*
la valeur de n qui est passée en parametre represente le numéro de la fiche que
l'on veut lire,
il faut donc aller lire dans l'index quelle est l'id de la n ième fiche ( il
faut donc ce déplacer de n-1 fois 5 caracteres dans l'index ), ensuite en
convertissant
le chiffre en entier on sait de combien on doit se déplacer dans le fichier file
pour trouver la fiche qui nous intéresse, l'id représentant le numéro logique
de chaque fiche dans le fichier file.
*/
void print(int n)
{
    char convert[5];
    int b,c;
    FICHE _aff;

    fseek(index,5,SEEK_SET); // on passe au dessus
// des 5 premiers caractere qui represente le last_id
    fseek(index, (n-1)*5,SEEK_CUR); // on se place de debut
// de l'id a lire
    b=fread(convert, sizeof(char[5]),1,index);
    if(b == 1) // la fonction fread a
// pu lire 1 tableau de 5 caracteres
    {
        c=atoi(convert); // on convertit
        fseek(file, (c-1)*sizeof(FICHE),SEEK_SET); // on se deplace pour
// arriver au debut de la fiche à lire
        fread(&_aff, sizeof(FICHE), 1, file); // on lit la fiche
        printf("\tnom : %s\n\tprenom : %s\n\tmontant : %s\n\n", _aff.name,
// _aff.firstname, _aff.money); // on affiche le resultat
    }
    else printf("erreur de lecture\n");

    printf("Appuyer sur une touche\n");
    getch();
}
// -----

```

```

// Fonction supprimer une fiche -----
/* Pour supprimer une fiche on va écraser son id dans l'index et remonter les
id suivant.

Pour se faire on effectue deux déplacements ( on pourrait les réduire à un seul
mais pour des
souplesse de clarté et de compréhension j'ai choisis de le faire en deux étapes.
L'opération de lecture qui
suit les déplacements pourrait aussi être évitée mais elle sert de protection,
car si l'utilisateur entre une valeur hors de l'index
l'opération de lecture échouera et ne renverra pas 5 et la fonction sera
avortée ( grâce au if ). Après ces opérations,
Le pointeur est donc à la fin du champ à effacer, on est prêt à lire le champ
suivant.
La boucle lit l'enregistrement suivant et écrase le précédent jusqu'à ce qu'on
arrive à la fin du fichier.
Attention avec cette technique le dernier id se retrouve deux fois dans le
fichier index.
C'est pourquoi on retrouve à la sortie de la boucle deux lignes de code qui
permettent d'écraser les derniers caractères par un EOF qui a le code ascii 26
*/
int delet(int n)
{
    int b=1,c=1,i;
    char convert[5];

    fseek(index,5,SEEK_SET); // on passe au dessus
    // des 5 premiers caractères qui représentent le last_id
    fseek(index,(n-1)*5,SEEK_CUR); // on se place au début
    // de l'id à écraser puis on le lit le pointeur se retrouve donc à la fin de l'id
    // à écraser
    c=fread(convert, sizeof(char), 5, index);

    if( c<5)
    {
        printf("la fiche n'existe pas\n");
        getch();
        return 0;
    }

    while(!feof(index))
    {
        b=fread(convert, sizeof(char), 5, index); // on lit l'id suivant

        if( b < 5) break;

        fseek(index,-10,SEEK_CUR); // on revient au début
        // de l'id que l'on veut écraser
        fwrite(convert, sizeof(char), 5, index);
        fseek(index,5,SEEK_CUR);
    }

    fseek(index,-5,SEEK_END); // on écrase le dernier
    // enregistrement par un EOF
    fprintf(index,"%c",26);
}

```

```

    printf("Appuyer sur une touche\n");
    getch();
    flush_fich(); // Voir la fin du fichier
pour comprendre l'utilisation de cette ligne.
    return 1;
}
// -----

// Fonction supprimer toutes les fiches -----
void delet_all()
{
    fclose(file); // il faut fermer les
fichiers avant de les supprimer sans quoi il est impossible de les supprimer.
    fclose(index);
    system("del C:\\index.txt"); // supprime le fichier
index.txt
    system("del C:\\fiche.txt"); // supprime le fichier
fiche.txt

    // maintenant il faut réouvrir les fichiers pour que l'utilisateur n'y voit
que du feu ;)

    file=fopen("C:\\fiche.txt","a");
    fclose(file);
    file=fopen("C:\\fiche.txt","r+");
    index=fopen("C:\\index.txt","a");
    fclose(index);
    index=fopen("C:\\index.txt","r+");

    last_id=0; // hé oui le programme ne sait pas qu'on a supprimer l'index et
qu'il faut recommencer à 0 il faut réinitialiser la variable last_id

    printf("Appuyer sur une touche");
    getch();
}
// -----

// Fonction de purge -----
/*
Le principe et de lire les fichier index et file, de reconstruire l'ordre des id
( la premiere fiche active reprend le numéro 1 )
et d'ecrire les fiches active avec leur nouvel id dans un fichier temporaire.
Une fois que les fichiers temporaire sont creer et remplis on supprime les
fichier index et file puis on renomme les fichier temporaire
en index.txt et file.txt :) Ainsi nous avons purger les fiches non-active et de
plus nous avons libéré des ids ce qui est important car le nombre
d'id est limité à 5 caractères soit une valeur maximale de 99999 fiches ( c'est
deja pas mal me direz-vous ;) mais comme les fiches non-active occupait elle
aussi des ids nous aurions pu arriver a la limite, la purge de la base de
données pourra etre une solution pour récupérer des ids adressables
*/
void flush_fich()
{

```

```

FILE * index_tmp;
FILE * file_tmp;

FICHE a;
char convert[5];
int i=0, _offset,b;

index_tmp=fopen("C:\\index_tmp.txt","w+");
file_tmp=fopen("C:\\fiche_tmp.txt","w+");

fseek(index,5,SEEK_SET); // on passe au dessus
des 5 premiers caracteres qui representent le last_id
itoa(i, a.id, 10);
fwrite(a.id,sizeof(char),5,index_tmp); // on ecrit une premiere
fois le last-id pour etre sure que sa place dans le fichier lui soit reservé

while(!feof(index))
{
    b=fread(convert, sizeof(char),5,index);
    if (b == 5)
    {
        _offset=atoi(convert);
        fseek(file,(_offset-1)*sizeof(FICHE),SEEK_SET); // -1 car
le last_id commence a 1, la premiere fiche correspond a un deplacement de 0.
        fread(&a, sizeof(FICHE),1,file);
        i++;
        itoa(i, a.id, 10); //
conversion du last_id en une chaine de caractere en base 10
        fwrite(&a,sizeof(FICHE),1,file_tmp);
        fseek(index_tmp,0,SEEK_END);
        fwrite(a.id,sizeof(char),5,index_tmp);
    }
}

last_id=i;
fseek(index_tmp,0,SEEK_SET); // on ecrit
le dernier id au debut du fichier il represente le last_id réel
fwrite(a.id,sizeof(char),5,index_tmp);

fclose(index);
fclose(file);
system("del C:\\index.txt"); //
supprime le fichier index.txt
system("del C:\\fiche.txt"); //
supprime le fichier fiche.txt

fclose(index_tmp);
fclose(file_tmp);
system("rename C:\\index_tmp.txt index.txt");
system("rename C:\\fiche_tmp.txt fiche.txt");

/*****/
//ouverture du fichier en mode add, si le fichier n'existe pas il est crée.

file=fopen("C:\\fiche.txt","a");
fclose(file);
// maintenant on peut ouvrir en r+, on est sure que le fichier existe.
file=fopen("C:\\fiche.txt","r+");
/*****/

```

```
/* *****  
// meme philosophie pour le fichier d'index  
index=fopen("C:\\index.txt","a");  
fclose(index);  
index=fopen("C:\\index.txt","r+");  
/* *****  
  
}  
// -----  
  
/* Avant de quitter la fonction delete on fait appel a la fonction  
fflush_fich() et ce  
pour enlever le caractère EOF a la fin de l'index car il reste une correction à  
apporter  
au programme. L'ajout d'une fiche apres qu'une fiche est été suprimé pose un  
probleme car  
l'écriture se fait apres le caractèere EOF. La solution temporaire que j'ai  
trouvé  
pour publier le document avant les examens est de purger avant d'écrire une  
nouvelle fiche  
;) Si l'ame vous en dit apportez donc la correction ou contactez moi a  
kfn_velour@hotmail.com */
```