COURS et TP DE LANGAGE C++

Chapitre 9

Les fichiers

Joëlle MAILLEFERT

joelle.maillefert@iut-cachan.u-psud.fr

IUT de CACHAN

Département GEII 2

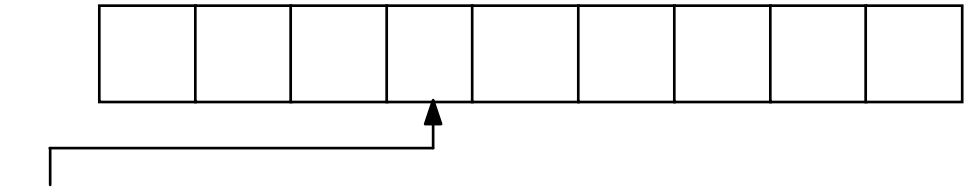
CHAPITRE 9

LES FICHIERS

GENERALITES

Un fichier est un ensemble d'informations stockées sur une mémoire de masse (disque dur, disquette, bande magnétique, CD-ROM).

Ces informations sont sauvegardées à la suite les unes des autres et ne sont pas forcément de même type (un char, un int, une structure ...)
Un **pointeur** permet de se repérer dans le fichier. On accède à une information en amenant le pointeur sur sa position.



pointeur

Sur le support de sauvegarde, le fichier possède un nom. Ce nom est composé de 2 parties : le nom proprement dit et l'extension. L'extension donne des informations sur le type d'informations stockées (à condition de respecter les extensions associées au type du fichier).

Exemples :

toto.txt	le fichier se nomme toto et contient du texte
mon_cv.doc	le fichier se nomme mon_cv et contient du texte, il a été édité sou WORD
ex1.cpp	le fichier se nomme ex1 et contient le texte d'un programme écrit en C++ (fichier source)
ex1.exe	le fichier se nomme ex1, il est exécutable
bibi.dll	le fichier se nomme bibi, c'est un fichier nécessaire à l'exécution d'un autre logiciel

Exercice IX_1 : Via l'explorateur de WINDOWS, reconnaître sur le disque dur du PC quelques fichiers.

On distingue généralement deux types d'accès:

1- Accès séquentiel (possible sur tout support, mais seul possible sur bande magnétique):

- Pas de cellule vide.
- On accède à une cellule quelconque en se déplaçant (via le pointeur sur un enregistrement du fichier), depuis la cellule de départ.
- On ne peut pas détruire une cellule.
- On peut par contre tronquer la fin du fichier.
- On peut ajouter une cellule à la fin.

2- Accès direct (RANDOM I/O) (Utilisé sur disques, disquettes, CD-ROM où l'accès séquentiel est possible aussi).

- Cellule vide possible.
- On peut directement accéder à une cellule.
- On peut modifier n'importe quelle cellule.

Il existe d'autre part deux façons de coder les informations stockées dans un fichier :

1- En binaire :

Fichier dit « binaire », les informations sont codées telles que. Ce sont en général des fichiers de nombres. Ils ne sont ni listables, ni éditables. Ils possèdent par exemple les extensions .OBJ, .BIN, .EXE, .DLL, .PIF etc …

Exercice IX_2 : Via le notepad ou l'éditeur de BC5, essayer d'éditer un fichier binaire.

2- en ASCII :

Fichier dit « texte », les informations sont codées en ASCII. Ces fichiers sont listables et éditables. Le dernier octet de ces fichiers est EOF (End Of File - caractère ASCII spécifique). Ils peuvent posséder les extensions .TXT, .DOC, .RTF, .CPP, .BAS, .PAS, .INI etc …

Exercice IX_3 : Via le notepad ou l'éditeur de BC5, essayer d'éditer quelques fichiers textes.

Un fichier possède des attributs, c'est à dire des droits d'accès : lecture, écriture (droit à modification), destruction etc…

Exercice IX_4 : Via le notepad, créer un fichier, y inscrire ce qui vous passe par la tête (1 ligne ou 2), le sauvegarder sous le nom **essai.dat** dans votre répertoire de travail, puis le fermer. Via l'explorateur de WINDOWS, et à l'aide du bouton droit de la souris, lire les attributs affectés par défaut par WINDOWS. Supprimer l'accès en écriture puis modifier le contenu du fichier et tenter une sauvegarde. Est-ce possible ?
Donner à nouveau l'accès en écriture et vérifier qu'une modification est possible.
Noter la taille du fichier fournie par WINDOWS et vérifier qu'elle correspond au nombre de caractères inscrits dans le fichier.

MANIPULATIONS GENERALES SUR LES FICHIERS

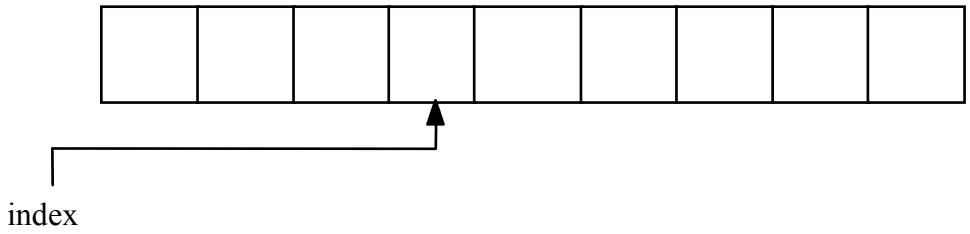
Opérations possibles avec les fichiers:
Créer - Ouvrir - Lire - Ecrire - Détruire – Renommer - Fermer.
La plupart des fonctions permettant la manipulation des fichiers sont rangées dans la bibliothèque standard STDIO.H, certaines dans la bibliothèque IO.H pour le BORLAND C++.
Ces fonctions sont très nombreuses. Seules quelques-unes sont présentées ici.

Pour manipuler un fichier, on commence toujours par l'ouvrir et vérifier qu'il est effectivement ouvert (s'il n'existe pas, cela correspond à une création).
Lorsque la manipulation est terminée, il faut fermer ce fichier et vérifier sa fermeture effective.

Le langage C++ ne distingue pas les fichiers à accès séquentiel des fichiers à accès direct, certaines fonctions de la bibliothèque livrée avec le compilateur permettent l'accès direct. Les fonctions standards sont des fonctions d'accès séquentiel.

1 - Déclaration: **FILE *index; // majuscules obligatoires pour FILE**

On définit un pointeur. Il s'agit du pointeur représenté sur la figure du début de chapitre. Ce pointeur repère une cellule donnée.



index est la variable qui permettra de manipuler le fichier dans le programme.

2 - Ouverture:

Il faut associer à la variable **index** au nom du fichier sur le support. On utilise la fonction **fopen** de prototype **FILE *fopen(char *nom, char *mode);**

On passe donc 2 chaînes de caractères
nom: celui figurant sur le support, par exemple: « a :\toto.dat »

mode (pour les fichiers TEXTES) :
« r » lecture seule
« w » écriture seule (destruction de l'ancienne version si elle existe)
« w+ » lecture/écriture (destruction ancienne version si elle existe)
« r+ » lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version.
« a+ » lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version, le pointeur est positionné à la fin du fichier.

mode (pour les fichiers BINAIRES) :
« rb » lecture seule
« wb » écriture seule (destruction de l'ancienne version si elle existe)
« wb+ »lecture/écriture (destruction ancienne version si elle existe)
« rb+ »lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version.
« ab+ »lecture/écriture d'un fichier existant (mise à jour), pas de création d'une nouvelle version, le pointeur est positionné à la fin du fichier.

A l'ouverture, le pointeur est positionné au début du fichier (sauf « a+ » et « ab+ », à la fin).

Exemple1 : **FILE *index ;**
index = fopen("a :\\toto.dat", "rb") ;

Exemple2 : **FILE *index ;**
char nom[30] ;
cout<< "Nom du fichier : " ;
cin >> nom ;
index = fopen(nom, "w") ;

3 - Fermeture:

On utilise la fonction de prototype **int fclose(FILE *f);**
Cette fonction retourne 0 si la fermeture s'est bien passée.

Exemple : **FILE *index ;**
index = fopen("a :\\toto.dat", "rb") ;
//
// Ici instructions de traitement
//
fclose(index) ;

Exercice IX_5 :

Ecrire un programme qui crée un fichier texte ("w") de nom **ex1.txt** dans le répertoire de travail, et qui le ferme.
Tester le programme puis vérifier la présence du fichier sur le disque dur. Quelle est sa taille ? Noter l'heure de création du fichier.

Exécuter à nouveau le programme et noter l'heure de création du fichier. A-t-elle changé ?

Remplacer l'attribut « w » par « r », exécuter le programme. L'heure de création du fichier a-t-elle changé ?

Sous WINDOWS, détruire le fichier.

Faire volontairement une faute de frappe dans le chemin d'accès au fichier, et exécuter le programme. Ceci provoque-t-il une erreur ?

Lorsqu'il y a un problème à l'ouverture du fichier, la fonction **fopen** retourne une valeur particulière du pointeur de fichier, la valeur NULL (ceci est une constante définie dans le fichier stdio.h). En testant la valeur retournée on peut ainsi réaliser un contrôle d'erreur :

Exemple : **FILE *index ;**
index = fopen("a :\\toto.dat", "rb") ;

if (index == NULL) cout << « Erreur a l'ouverture » ;
else
{
//
// Ici instructions de traitement

```
  //
fclose(index) ;
}
```

Exercice IX_6 : Modifier le programme de l'exercice IX_5 de sorte de vérifier si le chemin d'accès au fichier est correct.

4 - Destruction:

On utilise la fonction de prototype **int remove(char *nom);**
Cette fonction retourne 0 si la fermeture s'est bien passée.

Exemple1 : **remove(« a :\\toto.dat ») ;**

Exemple2 :
```
int x ;
x = remove("a :\\toto.dat") ;
if (x == 0) cout << « Fermeture OK : » ;
else cout << « Problème à la fermeture : » ;
```

5 - Renommer:

On utilise la fonction de prototype **int rename(char *oldname, char *newname);**
Cette fonction retourne 0 si la fermeture s'est bien passée.

Exemple1 : **rename("a :\\toto.dat" , "a :\\tutu.dat") ;**

Exemple2 :
```
int x ;
x = rename("a :\\toto.dat", "a :\\tutu.dat") ;
if (x == 0) cout << « Operation OK : » ;
else cout << "L'operation s'est mal passee : " ;
```

Exercice IX_7 : Modifier le programme de l'exercice IX_6 de sorte d'utiliser ces 2 dernières fonctions. Vérifier via l'explorateur de WINDOWS.

6 - Positionnement du pointeur au début du fichier:

On utilise la fonction de prototype **void rewind(FILE *f);**

Exemple :
```
FILE *index ;
index = fopen("a :\\toto.dat", "rb") ;  // pointeur au début

// ici traitement du fichier

rewind(index) ;   // repositionne le pointeur au début
```

7 - Fonction particulière aux fichiers à acces direct:

La fonction de prototype **int fseek(FILE *index , int offset , int direction)** déplace le pointeur de offset octets à partir de direction.
Valeurs possibles pour direction:

0 -> à partir du début du fichier.
1 -> à partir de la position courante du pointeur.
2 -> en arrière, à partir de la fin du fichier.

Cette fonction retourne « offset » si la manipulation s'est bien passée , retourne 0 si le pointeur n'a pu être déplacé.

Exemple : **FILE *index ;**
index = fopen("a :\\toto.dat", "rb") ; // pointeur au début

// ici manipulation du fichier

fseek(index, 5, 1) ; // déplace le pointeur de 5 position à partir de la
// position courante du pointeur

MANIPULATIONS DES FICHIERS TEXTES

1- Ecriture dans le fichier:

La fonction de prototype **int putc(char c, FILE *index)** écrit la valeur de c à la position courante du pointeur, le pointeur avance d'une case mémoire.
Cette fonction retourne –1 en cas d'erreur.

Exemple : **putc('A', index) ;**

La fonction de prototype **int fputs(char *chaîne, FILE *index)** est analogue avec une chaîne de caractères. Le pointeur avance de la longueur de la chaîne ('\0' n'est pas rangé dans le fichier).
Cette fonction retourne le code ASCII du caractère, retourne -1 en cas d'erreur (par exemple tentative d'écriture dans un fichier ouvert en lecture)

Exemple : **fputs("BONJOUR ! ", index) ;**

Exercice IX_8 : Modifier le programme de l'exercice IX_5 : Ouvrir le fichier, saisir quelques caractères, les ranger dans le fichier, puis saisir une chaîne de caractères et la ranger dans le fichier. Ne pas faire de contrôle d'erreur Vérifier via le notepad.

Ouvrir maintenant le fichier en mode "a". Exécuter le programme. Vérifier via le notepad que le fichier a bien été modifié.

Exercice IX_9 : Ouvrir maintenant le fichier en mode « r » et exploiter le contrôle d'erreur.

La fonction de prototype **int putw(int n, FILE *index)** écrit la valeur de n (codée en ASCII) à la position courante du pointeur, le pointeur avance d'une case mémoire.
Cette fonction retourne –1 en cas d'erreur.

Exemple : **int n = 45 ;**
putw(n, index) ;

2- Relecture d'un fichier:

Les fichiers texte se terminent par le caractère ASCII EOF (de code –1). Pour relire un fichier, on peut donc exploiter une boucle jusqu'à ce que la fin du fichier soit atteinte.

La fonction de prototype **int getc(FILE *index)** lit 1 caractère, et retourne son code ASCII, sous forme d'un entier. Cet entier vaut –1 (EOF) en cas d'erreur ou bien si la fin du fichier est atteinte. Via une conversion automatique de type, on obtient le caractère.

Exercice IX_10 (à expérimenter) :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

    void main()
   {
      FILE *index;
      char c=0   ; // initialisation pour le 1er tour

      index = fopen("c:\\bc5\\sources\\ex1.txt","r");
      while(c!=EOF)
      {
         c=getc(index); // on utilise une conversion de type automatique
         cout<<c;
      }
      fclose(index);

      cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE ";
      getch();
   }
```

Exercice IX_11 : Copier un fichier dans un autre, vérifier via le notepad.

Exercice IX_12: Calculer et afficher le nombre de caractères d'un fichier texte.
Ecrire ensuite une fonction de prototype **int taille(char *nom)** qui retourne la taille de ce fichier et la mettre en œuvre dans le programme principal. Le fichier doit être ouvert et fermé dans la fonction. Le paramètre **nom** désigne le nom du fichier sur le disque dur. Il doit être fourni dans le programme principal et passé en paramètre.

La fonction de prototype **char *fgets(char *chaine, int n, FILE *index)** lit n-1 caractères à partir de la position du pointeur et les range dans chaîne en ajoutant '\0'. Retourne un pointeur sur la chaîne, retourne le pointeur NULL en cas d'erreur, ou bien si la fin du fichier est atteinte.

Exemple :
```
FILE *index ;
char texte[10] ;
// ouverture
fgets(texte, 7, index) ;  // lit 7 caractères dans le fichier et forme la chaîne
                          // « texte » avec ces caractères
```
La fonction de prototype **int getw(FILE *index)** lit 1 nombre stocké sous forme ASCII dans le fichier, et le retourne. Cet entier vaut –1 en cas d'erreur ou bien si la fin du fichier est atteinte.

MANIPULATIONS DES FICHIERS BINAIRES

La fonction **int feof(FILE *index)** retourne 0 tant que la fin du fichier n'est pas atteinte.

La fonction **int ferror(FILE *index)** retourne 1 si une erreur est apparue lors d'une manipulation de fichier, 0 dans le cas contraire.

La fonction de prototype **int fwrite(void *p, int taille_bloc, int nb_bloc, FILE *index)** écrit à partir de la position courante du pointeur **index** nb_bloc X taille_bloc octets lus à partir de l'adresse p. Le pointeur fichier avance d'autant.
Le pointeur p est vu comme une adresse, son type est sans importance.
Cette fonction retourne le nombre de blocs écrits (0 en cas d'erreur, ou bien si la fin du fichier est atteinte).

Exemple: taille_bloc = 4 (taille d'un entier en C++), nb_bloc=3, écriture de 3 entiers.
```
int tab[10] ;
fwrite(tab,4,3,index) ;
```

La fonction de prototype **int fread(void *p,int taille_bloc,int nb_bloc,FILE *index)** est analogue à **fwrite** en lecture.
Cette fonction retourne le nombre de blocs luts (0 en cas d'erreur, ou bien si la fin du fichier est atteinte).

Exercice IX_13: Créer et relire un fichier binaire de 3 entiers.
Ecrire ensuite une fonction de prototype **void creer(char *nom)** qui crée le fichier de 10 entiers et la mettre en œuvre dans le programme principal. Le fichier doit être ouvert et fermé dans la fonction. Le paramètre **nom** désigne le nom du fichier sur le disque dur. Il doit être fourni dans le programme principal et passé en paramètre.

Ecrire de même une fonction de prototype **void lire(char *nom)** qui relie le fichier et affiche son contenu. La mettre en œuvre dans le programme principal. Le paramètre **nom** désigne le nom du fichier sur le disque dur. Il doit être fourni dans le programme principal et passé en paramètre.

Ecrire maintenant une fonction de prototype **void ajout(char *nom, int n)** qui ajoute l'entier n au fichier précédent. La mettre en œuvre dans le programme principal. Le paramètre **nom** désigne le nom du fichier sur le disque dur. Il doit être fourni dans le programme principal et passé en paramètre. Relire ce fichier grâce à la fonction **void lire(char *nom).**

Ecrire maintenant une fonction de prototype **int cherche(char *nom, int n)** qui recherche si l'entier n existe dans le fichier précédent et relire le fichier grâce à la fonction **void lire(char *nom).** Cette fonction retourne la position du nombre si il existe, 0 sinon. La mettre en œuvre dans le programme principal. Le paramètre **nom** désigne le nom du fichier sur le disque dur. Il doit être fourni dans le programme principal et passé en paramètre.

Exercice IX_14: Créer une structure nom, prénom, âge. Ecrire un programme de gestion de fichier (binaire) avec menu d'accueil: possibilité de créer le fichier, de le lire, d'y ajouter une fiche, d'en rechercher une.

CORRIGE DES EXERCICES

Exercice IX_1:

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

    void main()
   {
      FILE *index;
      index = fopen("c:\\bc5\\sources\\ex1.txt","w");
      fclose(index);
      cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE "; getch();
   }
```

Exercice IX_2 :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

    void main()
   {
      FILE *index;
      index = fopen("c:\\bc5\\sources\\ex1.txt","w");
      if(index == NULL) cout << "Erreur dans le chemin d'accès\n";
      else
      {
         cout << "Création du fichier OK\n";
         fclose(index);
      }
      cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE "; getch();
   }
```

Exercice IX_5 :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

    void main()
   {
      FILE *index = fopen("c:\\bc5\\sources\\ex1.txt","w");
      fclose(index);
      cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE "; getch();
   }
```

Exercice IX_6 :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

    void main()
   {
      FILE *index = fopen("c:\\bc5\\sources\\ex1.txt","w");
      if(index == NULL) cout << "Erreur dans le chemin d'accès\n";
      else
      {
         cout << "Création du fichier OK\n";
         fclose(index);
      }
      cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE ";
      getch();
      fclose(index);
   }
```

Exercice IX_7 :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

    void main()
   {
      FILE *index = fopen("c:\\bc5\\sources\\ex1.txt","w");
      if(index == NULL) cout << "Erreur dans le chemin d'accès\n";
      else
      {
         cout << "Création du fichier OK\n";
         fclose(index);
      }
      rename( "c:\\bc5\\sources\\ex1.txt", "c:\\bc5\\sources\\change.txt");
      remove( "c:\\bc5\\sources\\ex1.txt");
      cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE ";
      getch();
      fclose(index);
   }
```

Exercice IX_8 :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

    void main()
   {
      FILE *index = fopen("c:\\bc5\\sources\\ex1.txt","w");
      char c, texte[20];
      for(int i=0; i<5; i++)
      {
         cout<<"Saisir un caractere :"; cin>> c;
         putc(c, index);
      }
      cout<<"Votre message :"; cin>>texte;
      fputs(texte,index);
      fclose(index);
      cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE ";
      getch();
   }
```

Exercice IX_9 :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

    void main()
   {
      FILE *index = fopen("c:\\bc5\\sources\\ex1.txt","r");
      char c   ;
      int  n;
      index = fopen("c:\\bc5\\sources\\ex1.txt","r");
      for(int i=0;i<3;i++)
      {
         cout<<"Saisir un caractere :"; cin>> c;
         n=putc(c, index);
         cout<<"n="<<n<<"\n";
      }
      fclose(index);
      cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE ";
      getch();
   }
```

Exercice IX_11 :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

    void main()
   {
      FILE *index1, *index2;
      char c=0; // initialisation pour le 1er tour
      index1 = fopen("c:\\bc5\\sources\\ex1.txt","r");
      index2 = fopen("c:\\bc5\\sources\\ex2.txt","w");

      while(c!=EOF)
      {
         c=getc(index1);
         putc(c, index2);
      }
      fclose(index1);
      fclose(index2);

      cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE "; getch();
   }
```

Exercice IX_12 :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

    int taille(char *nom)
   {
      FILE *index1 = fopen(nom,"r");
      int n=0;
      char c=0; // initialisation pour le 1er tour
      while(c!=EOF){
         c=getc(index1);
         n++;
      }
      fclose(index1);
      return n;
   }

    void main()
   {
      char mon_fichier[50]= "c:\\bc5\\sources\\ex1.txt";
      int resultat= taille(mon_fichier);
      cout<<"Taille du fichier :"<<resultat<<"\n";
      cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE "; getch();
   }
```

Exercice IX_13 :

```
#include <stdio.h>
#include <conio.h>
#include <iostream.h>

    void creer(char *nom)
   {
      FILE *index = fopen(nom,"wb"); int n;
      for(int i=0;i<3;i++)
      {
         cout<<"Saisir un nombre: "; cin >> n ;
         fwrite(&n,4,1,index);
      }
      fclose(index);
   }

    void lire (char *nom)
   {
      FILE *index = fopen(nom,"rb"); int n, u=0  ;
      cout<<"Voici le contenu du fichier:\n";
      while(u==0)
      {
         fread(&n,4,1,index); u = feof(index);
         if(u==0)cout<< n <<" ";
      }
      cout<<"\n\n"; fclose(index);
   }

    void ajout (char *nom, int n)
   {
      FILE *index = fopen(nom,"ab+");
      fwrite(&n,4,1,index); fclose(index);
   }

    int cherche(char *nom, int n)
   {
      FILE *index = fopen(nom,"rb"); int place=0, u=0, n_lu, trouve=0;
      while(u==0)
      {
         fread(&n_lu,4,1,index); place++;
         if(n== n_lu) trouve = place;  // trouve le dernier
         u=feof(index);
      }
      fclose(index);
      return trouve;
   }

    void main()
   {
      char mon_fichier[50] = "c:\\bc5\\sources\\essai.dat";
      int plus, combien, numero;
      creer(mon_fichier); lire(mon_fichier);
      cout<<"Saisir le nombre à ajouter : "; cin>>plus;
      ajout(mon_fichier,plus); lire(mon_fichier);
      cout<<"Saisir le nombre a rechercher : "; cin>>combien;
      numero = cherche(mon_fichier, combien);
      if(numero==0)cout<<"Ce nombre n'existe pas\n";
      else cout<<"Ce nombre se trouve a la place numero "<<numero<<"\n";
      cout<<"\nPOUR SORTIR FRAPPER UNE TOUCHE "; getch();
   }
```

Exercice IX_14 :

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
#include <stdio.h>

   typedef
       struct
      {
         char nom[10];
         char prenom[10];
         int age;
      } carte;// création d'un type carte

    void creer_fichier(char *nom)
   {
      FILE *index = fopen(nom,"wb"); char choix; carte fiche;
      clrscr(); cout<<"CREATION DU FICHIER \n\n";
      do
      {
         cout<<"\nSAISIE D'UNE FICHE ?(o/n) "; choix = getch();
         if ((choix=='o')||(choix=='O'))
         {
            cout<<"\nNOM: ";  cin>>fiche.nom;
            cout<<"PRENOM: "; cin>>fiche.prenom;
            cout<<"AGE: ";    cin>>fiche.age;
            fwrite(&fiche,sizeof(carte),1,index);
         }
      }
      while((choix=='o')||(choix=='O'));
      fclose(index);
   }

    void lire_fichier(char *nom)
   {
      FILE * index = fopen(nom,"rb"); carte fiche; int compteur=0;
      clrscr(); cout<<"LECTURE DU FICHIER\n\n";
      if (index == NULL) cout<<"\nERREUR, CE FICHIER N'EXISTE PAS\n\n";
      else
      {
         cout<<"\nLISTING DU FICHIER\n\n";
         while(fread(&fiche,sizeof(carte),1,index)!=0)
         {
            cout<<"fiche numero "<<compteur<<" : "; compteur++;
            cout<<fiche.nom<<" "<<fiche.prenom<<" "<<fiche.age<<"\n";
         }
         fclose(index);
      }
      cout<<"POUR CONTINUER FRAPPER UNE TOUCHE "; getch();
   }
```

Suite page suivante :

```
 void recherche(char *nom)
{
   FILE *index = fopen(nom,"rb");
   carte fiche;
   int compteur=0;
   char trouve = 0, nn[10], pp[10];

   clrscr();
   cout<<"RECHERCHE DE FICHE\n\n";

   cout<<"\nFICHE A RETROUVER:\n";
   cout<<"NOM: ";   cin>>nn;
   cout<<"PRENOM: ";cin>>pp;

   while((fread(&fiche,sizeof(carte),1,index)!=0)&&(trouve==0))
   {
      if((strcmp(fiche.nom,nn)==0)&&(strcmp(fiche.prenom,pp)==0))
      {
         trouve=1;
         cout<<"FICHE RETROUVEE, NUMERO:"<<compteur<<"\n";
      }
      compteur++;
   }
   if (trouve==0)cout<<"CETTE FICHE N'EXISTE PAS\n";
   fclose(index);
   cout<<"POUR CONTINUER FRAPPER UNE TOUCHE ";
   getch();
}
```

Suite page suivante :

```
 void main()
{
   FILE *fichier;
   char mon_fichier[50]= "c:\\bc5\\sources\\ex14.dat";
   char choix;
   do
   {
      clrscr();
      cout<<"\t\t\tGESTION DE FICHIER\n";
      cout<<"\t\t\t------------------\n\n\n";
      cout<<"CREATION DU FICHIER  ---> 1\n";
      cout<<"LECTURE DU FICHIER   ---> 2\n";
      cout<<"AJOUTER UNE FICHE    ---> 3\n";
      cout<<"RECHERCHER UNE FICHE ---> 4\n";
      cout<<"SORTIE     ---> S\n\n";
      cout<<"VOTRE CHOIX: ";
      choix = getch();
      switch(choix)
      {
         case '1':creer_fichier(mon_fichier);
            break;
         case '2':lire_fichier(mon_fichier);
            break;
         case '3':ajout(mon_fichier);
            break;
         case '4':recherche(mon_fichier);
            break;
      }
   }
   while ((choix!='S') && (choix!='s'));
}
```