1.3 GESTION DE FICHIERS

Dans les exercices qui suivent, on utilisera la fonction de la bibliothèque standard : *int feof(FILE* f)* qui retourne EOF ou 0 suivant que le fichier f (supposé ouvert en lecture) est ou non en dernière position.

Exercice 29 Soit un fichier de données structuré en une suite de lignes contenant chacune un nom de personne, un nom de pièce, un nombre et un prix. Exemple :

```
Dupons Villebrequin 10 1000
Durant Brosse       20 567
```

Ecrire une procédure *main* dans laquelle on déclarera les variables suivantes :

- *cNom* et *cArticle* : tableaux de 80 caractères,
- *iNombre* et *iPrix* : entiers.

Le corps de la procédure consistera en une boucle dont chaque itération lira une ligne et l'imprimera. La lecture se fera par un appel à *fscanf* affectant les 4 champs de la ligne aux 4 variables *cNom*, *cArticle*, *iNombre* et *iPrix*. L'écriture consistera à imprimer *cNom*, *cArticle* et le produit $iNombre * iPrix$.

Exercice 30 Soit un fichier de données identiques à celui de l'exercice précédent. Ecrire une procédure *main* qui :

1. Lise le fichier en mémorisant son contenu dans un tableau de structures, chaque structure permettant de mémoriser le contenu d'une ligne (nom, article, nombre et prix).
2. Parcoure ensuite ce tableau en imprimant le contenu de chaque structure.

Exercice 31 Faire le même exercice que le précédent, mais en mémorisant le fichier de données, non pas dans un tableau de structures, mais dans une liste de structures chaînées.

Exercice 32 Modifier le programme précédent tel que :

1. En écrivant la procédure d'impression d'une structure *commande*, procédure qui admettra en paramètre un pointeur vers une telle structure.
2. En écrivant une fonction de recherche de commande maximum (celle pour laquelle le produit nombre∗prix est maximum). Cette fonction admettra en paramètre un pointeur vers la structure *commande* qui est en tête de la liste complète, et rendra un pointeur vers la structure recherchée.
3. Le programme principal sera modifié de manière à faire appel à la fonction de recherche de commande maximum et à imprimer cette commande.

EXERCICE 29

```
#include "stdio.h"
/*****************************************************************************/
/*                              programme principal                          */
/*            Lecture et impression du contenu d'un fichier                  */
/*                       d'enregistrements de commandes                      */
/*****************************************************************************/
void main()
{
  /* Déclaration du fichier de données*/
  FILE* FCommandes;

  /* Déclaration des champs de la commande */
  char cNom[80];
  char cArticle[80];
  int iNombre,iPrix;

   /* Ouverture du fichier */
   FCommandes = fopen("Td2_exo11.data","r");

   /* Si le fichier ne peut pas s'ouvrir en lecture */
   if (FCommandes == NULL)
      /* Affichage d'un message d'erreur */
      printf("Impossible d'ouvrir le fichier Td2_exo11.data\n");

   /* Si le fichier peut s'ouvrir en lecture */
   else
   {
     /* Tant qu'on est pas à la fin du fichier, on lit les enregistrements */
     while(fscanf(FCommandes,"%s %s %d %d",cNom,cArticle,&iNombre,&iPrix) != EOF)
       /* Affichage du contenu du fichier */
      printf("%s %s %d\n",cNom,cArticle,iNombre * iPrix);

      /* Fermeture du fichier*/
      fclose(FCommandes);

   } /* Fin du else*/
}
```

EXERCICE 30

```
#include <stdio.h>

/*************************************************************************/
/*                              programme principal                      */
/*            Lecture et impression du contenu d'un fichier              */
/*                      d'enregistrements de commandes                   */
/*                          a l'aide d'une structure                     */
/*************************************************************************/
void main()
{ /* Déclaration du fichier de données*/
  FILE* FCommandes;

  /* Définition d'une structure de commande */
  struct commande
   { char cNom[80];
     char cArticle[80];
     int iNombre,iPrix;
   };

  /* Déclaration d'une constante nombre de commande max */
  const int iNB_COM = 100;

  /* Déclaration d'un tableau de commandes */
  struct commande tab_com[iNB_COM];

  /* Déclaration d'un index pour le tableau */
  int iIndex;

  /* Déclaration d'un index pour le dernier élément valide dans le tableau */
  /* après remplissage                                                     */
  int iDerniere;

  /* Si le fichier ne peut pas s'ouvrir en lecture */
  if ((FCommandes = fopen("Td2_exo11.data","r")) == NULL)
      /* Affichage d'un message d'erreur */
      printf("Impossible d'ouvrir le fichier Td2_exo11.data\n");
  /* Si le fichier peut s'ouvrir en lecture */
  else
  {
   /* Boucle de lecture des commandes   */

   /* Initialisation de l'index */
   iIndex = 0;
```

```
   /* tant que l'index est inférieur au nombre max de commandes et   */
   /*  qu'on est pas à la fin du fichier, on lit les enregistrements */
   while(iIndex < iNB_COM && fscanf(FCommandes,"%s %s %d %d",
                             tab_com[iIndex].cNom,
                             tab_com[iIndex].cArticle,
                             &tab_com[iIndex].iNombre,
                             &tab_com[iIndex].iPrix) != EOF)

      /* Incrémentation de l'index */
      iIndex++;

   /* S'il y a plus de 100 commandes */
   if (iIndex >= iNB_COM)
      printf("le tableau tab_com est sous-dimentionné\n");

   /* S'il y a moins de 100 commandes */
   else
      {
      /* Impression des commandes mémorisées   */

      /* Initialisation de l'index de la dernière commande */
      iDerniere = iIndex - 1;

      /* Pour toutes les Commandes */
      for (iIndex = 0; iIndex <= iDerniere; iIndex++)
         /* Affichage des commandes */
         printf("%s %s %d %d\n", tab_com[iIndex].cNom, tab_com[iIndex].cArticle,
                                 tab_com[iIndex].iNombre, tab_com[iIndex].iPrix);

      /* Fermeture du fichier */
      fclose(FCommandes);
      }
   }
}
```

EXERCICE 31

```
#include <stdio.h>
#include <stdlib.h>

/*************************************************************************/
/*                              programme principal                      */
/*            Lecture et impression du contenu d'un fichier              */
/*                      d'enregistrements de commandes                   */
/*                          a l'aide d'une liste chaînée                 */
/*************************************************************************/
void main()
{
  /* Déclaration du fichier de données*/
  FILE* FCommandes;

  /* Définition d'une structure de commande */
  struct commande
   { char cNom[80];
     char cArticle[80];
     int iNombre,iPrix;
     /* Pointeur sur l'élément suivant de la liste chaînée */
     struct commande *suiv;
   };

  /* Déclaration et initialisation de la liste de commandes */
  struct commande *l_com = NULL;
  /* Déclaration de pointeur sur la commande courante et la commande précédente */
  struct commande *prec,*cour;
  /* Déclaration de la valeur de retour du scanf */
  int iVal_ret;

  /* Ouverture du fichier */
   FCommandes = fopen("Td2_exo11.data","r");
  /* Si le fichier ne peut pas s'ouvrir en lecture */
  if (FCommandes == NULL)
      /* Affichage d'un message d'erreur */
      printf("Impossible d'ouvrir le fichier Td2_exo11.data\n");
  /* Si le fichier peut s'ouvrir en lecture */
  else
  {
   /* Création de la liste chaînée de commandes */

   /* Boucler tant qu'il y a des lignes dans le fichier */
   do
   {
     /* Allocation mémoire du pointeur sur la commande courante */
     cour = (struct commande*)malloc(sizeof(struct commande));
```

```
      /* Lecture de la ligne de commande */
      iVal_ret = fscanf(FCommandes,"%s %s %d %d",
                              cour -> cNom,
                              cour -> cArticle,
                              &(cour -> iNombre),
                              &(cour -> iPrix));

      /* Si on est à la fin du fichier */
      if (iVal_ret == EOF)
      { /* Libération de la mémoire occupée par le pointeur sur la commande */
        /* courante                                                          */
        free(cour);

        /* Si la liste est non vide, alors le suivant de l'élément précédent */
        /* est nul                                                           */
        if(l_com != NULL) prec -> suiv = NULL;
       } /* Fin du if (iVal_ret == EOF)

       /* Si on est pas à la fin du fichier */
       else
       {  /* Si la liste est vide, alors la liste contient la commande courante */
          if (l_com == NULL) l_com = cour;
          /* Si la liste est non vide, le suivant de l'élément précédent est */
          /* l'élément courant                                               */
          else prec -> suiv = cour;
          /* L'élément précédent devient l'élément courant */
          prec = cour;
        } /* Fin du else de if (iVal_ret == EOF) */
   }
   while (iVal_ret != EOF); /* Tant qu'on est pas à la fin du fichier */


   /* Parcours de la liste avec impression   */
   /* Si la liste est vide */
   if (l_com == NULL)
      printf("La liste de commandes est vide\n");
   /* Si la liste est non vide */
   else
   {
      /* Pour tous les éléments de la liste */
      for (cour = l_com; cour != NULL; cour = cour -> suiv)
        /* Affichage de l'élément courant */
        printf("%s %s %d %d\n",
                cour -> cNom,cour -> cArticle,cour -> iNombre,cour -> iPrix);
   }

   /* Fermeture du fichier   */
   fclose(FCommandes);
   }
}
```

EXERCICE 32

```
#include <stdio.h>
#include <stdlib.h>

/* Type commun a toutes les procédures    */
/* Définition d'une structure de commande */
  struct commande
   { char cNom[80];
     char cArticle[80];
     int iNombre,iPrix;
     /* Pointeur sur l'élément suivant de la liste chaînée */
     struct commande *suiv;
   };


/* Fonction qui imprime une structure commande                      */
/* Paramètres d'entrée : struct commande *com : une commande        */
/* Paramètres de sortie : rien                                      */
void print_com(struct commande *com)
{
printf("%s %s %d %d\n",
       com -> cNom,com -> cArticle,com -> iNombre,com -> iPrix);
}

/* Fonction qui recherche la commande pour laquelle le produit             */
/* nombre * prix est le maximum                                            */
/* Paramètres d'entrée : struct commande *l_com : la liste des commandes   */
/* Paramètres de sortie : struct commande *: un pointeur vers la structure */
/*                        commande recherchée ou NULL si l_com est vide    */
struct commande *max_com(struct commande * l_com)
{
  /* Déclaration d'un pointeur sur la commande pour laquelle le produit  */
  /* nombre * prix est le maximum maximum                                */
  struct commande *pmax;
  /* Déclaration d'un pointeur sur la commande courante */
  struct commande *cour;
  /* Déclaration des valeur maximum et courante */
  int iVmax,iVcour;

  /* SI la liste est vide */
  if (l_com == NULL)
   /* On sort de la fonction et on retourne NULL */
   return(NULL);
```

```
  /* Si la liste est non vide */
  else
  { /* Le pointeur sur la commande max est initialisé à la 1ère commande */
    pmax = l_com;
    /* Initialisation de la valeur max */
    iVmax = (pmax -> iNombre) * (pmax -> iPrix);

    /* Pour toutes les commandes */
    for (cour = l_com -> suiv; cour != NULL; cour = cour ->suiv)
    { /* Initialisation de la valeur courante */
      iVcour = (cour -> iNombre) * (cour -> iPrix);
      /* Si la valeur courante est supérieure à la valeur max */
      if (iVcour > iVmax)
      { /* Le max est modifié */
        iVmax = iVcour;
        pmax = cour;
      } /* Fin du if (iVcour > iVmax)*/
    }/* Fin du for */

    /* On retourne l'élément maximum */
    return(pmax);
  } /* Fin de  else*/
}

/*************************************************************************/
/*                            programme principal                        */
/*************************************************************************/
void main()
{
  /* Déclaration du fichier de données*/
  FILE* FCommandes;
  /* Déclaration et initialisation de la liste  de commandes */
  struct commande *l_com = NULL;
  /* Déclaration de pointeurs sur la commande courante et la commande précédente */
  struct commande *prec,*cour;
  /* Déclaration de la valeur de retour du scanf */
  int iVal_ret;

  /* Ouverture du fichier */
   FCommandes = fopen("Td2_exo11.data","r");
  /* Si le fichier ne peut pas s'ouvrir en lecture */
  if (FCommandes == NULL)
      /* Affichage d'un message d'erreur */
      printf("Impossible d'ouvrir le fichier Td2_exo11.data\n");
```

```
/* Si le fichier peut s'ouvrir en lecture */
else
{ /* Création de la liste chaînée de commandes */

  /* Boucler tant qu'il y a des lignes dans le fichier */
  do
   { /* Allocation mémoire du pointeur sur la commande courante */
    cour = (struct commande*)malloc(sizeof(struct commande));

    /* Lecture de la ligne de commande */
    iVal_ret = fscanf(FCommandes,"%s %s %d %d",
                           cour -> cNom,
                           cour -> cArticle,
                           &(cour -> iNombre),
                           &(cour -> iPrix));

     /* Si on est à la fin du fichier */
     if (iVal_ret == EOF)
     {
       /* Libération de la mémoire occupée par le pointeur sur la commande */
       /* courante                                                         */
       free(cour);

       /* Si la liste est non vide, alors le suivant de l'élément précédent */
       /* est nul                                                           */
       if(l_com != NULL) prec -> suiv = NULL;
      } /* Fin du if (iVal_ret == EOF)

      /* Si on est pas à la fin du fichier */
      else
      {
      /* Si la liste est vide, alors la liste contient la commande courante*/
       if (l_com == NULL) l_com = cour;
       /* Si la liste est non vide, le suivant de l'élément précédent est */
       /* l'élément courant                                               */
       else prec -> suiv = cour;
       /* L'élément précédent devient l'élément courant */
       prec = cour;
      } /* Fin du else de if (iVal_ret == EOF) */
  }
  /* Tant qu'on est pas à la fin du fichier */
  while (iVal_ret != EOF);

 /* Parcours de la liste avec impression   */

 /* Si la liste est vide */
 if (l_com == NULL)
    printf("La liste de commandes est vide\n");
```

```
   /* Si la liste est non vide */
   else
   {
      /* Pour tous les éléments de la liste */
      for (cour = l_com; cour != NULL; cour = cour -> suiv)
        /* Affichage de l'élément courant */
        print_com(cour);

      /* Recherche et impression de la commande maximum   */
      printf("La commande maximum est :\n");
      print_com(max_com(l_com));
      }

   /* Fermeture du fichier   */
   fclose(FCommandes);
   }
}
```

EXERCICE 33

```
#include<stdio.h>
#include<stdlib.h>

/* Déclaration d'un type élément de liste chaînée */
typedef struct lis
{
 /* Valeur de l'élément */
 int iValeur;
 /* Pointeur sur l'élément suivant */
 struct lis* suivant;
} *element;

/* Déclaration d'un type liste chaînée */
typedef struct t
{
 /* pointeurs sur le premier élément de la liste, sur le dernier */
 /* et sur l'élément courant                                     */
 element prem,der,cour;
} *liste;

/* Fonction de création d'une liste */
/* Paramètres d'entrée : Aucun      */
/* Paramètre de sortie : une liste  */
liste creer_liste()
{
  /* Déclaration d'une liste locale */
  liste Liste_Entier;

  /* Allocation mémoire de la liste */
  Liste_Entier = (liste)malloc(sizeof(struct t));
```