UNIVERSITE PARIS DAUPHINE

2003 - 2004

Université Paris Dauphine

IUP Génie Mathématique et Informatique

2ème année

MISE A NIVEAU INFORMATIQUE

LANGAGE C - EXEMPLES DE PROGRAMMES

Maude Manouvrier

TABLE DES MATIERES

RECHERCHE DICHOTOMIQUE DANS UN TABLEAU D'ENTIERS

```
#include<stdio.h>

/* Programme de recherche dichotomique d'un élément dans une liste d'entiers  */
int main()
{

  /* DECLARATION DES VARIABLES */

  int iTableau[]={1,2,3,5,6,8,9};  /* Tableau TRIE d'entiers */
  int iRecherche; /* Elément recherché                                        */
  int iPremier;   /* Indice du premier élément du sous-tableau analysé        */
  int iDernier;   /* Indice du dernier élément du sous-tableau analysé        */
  int iMilieu;    /* Indice de l'élément du milieu du sous-tableau analysé    */
  int iTrouve;    /* Booléen indiquant si l'élément est trouvé                */
  int iFin=1;     /* Indication de fin de saisie (0=fin)                      */

  /* Tant que l'utilisateur souhaite faire des recherches */
  while(iFin)
    {
      printf("Quel élément recherchez-vous ? ");
      scanf("%d",&iRecherche);

      /* Initialisation des variables*/
      iPremier=0;
      iDernier=6;
      iTrouve=0;

      /* Tant qu'on a pas trouve l'élément recherché ou que le sous-tableau */
      /* contient plus de 1 élément                                         */
      while((iPremier <= iDernier)&&(iTrouve==0))
      {
        /* Calcul de la position de l'élément du milieu */
        iMilieu=(iPremier+iDernier)/2;

        /* Si l'élément du milieu est l'élément recherché */
        if(iTableau[iMilieu]==iRecherche) iTrouve =1;
        else
          {
            /* Si la valeur recherchée est plus petite */
            /* que la valeur du l'élément du milieu */
            /* Alors on regarde le sous-tableau de gauche */
            if(iTableau[iMilieu]>iRecherche) iDernier = iMilieu -1;
            /* sinon on regarde le sous-tableau de droite*/
            else iPremier = iMilieu +1;
          }
      }

      if(!iTrouve) printf("Cette valeur n'appartient pas à la liste\n");
      else printf("Cette valeur appartient à la liste\n");

      printf("Voulez-vous continuer ? (Taper 0 pour sortir du programme) : ");
      scanf("%d",&iFin);

      /* Si l'utilisateur ne saisit pas un nombre, on sort du programme */
      if(!isalpha(iFin)) iFin=0;

      /* reprise d'une recherche */
      iTrouve=0;

    } /* Fin du while */

} /* Fin du main */
```

TRI PAR SELECTION ORDINAIRE D'UN TABLEAU D'ENTIERS

```
#include<stdio.h>

/* Programme qui trie les éléments d'une liste d'entiers */
/* (tri par sélection ordinaire)                         */

const int TAILLE=10;

int main()
{

/* DECLARATION DES VARIABLES */

int iTableau[TAILLE]; /* Liste d'entiers de taille max TAILLE avec i<TAILLE      */
int iIndiceElCour;    /* L'indice de l'élément courant                           */
int iMin;             /* L'élément de valeur minimum dans la liste               */
int iCompteur;        /* Un compteur pour la boucle                              */
int iEchange;         /* Une variable temporaire pour l'échange de deux éléments */


/* CORPS DU PROGRAMME */

/* Saisie des éléments de la liste */
for(iIndiceElCour =0; iIndiceElCour <TAILLE; iIndiceElCour ++)
 {
   /* Affichage d'un message demandant la valeur du ième entier */
   /* avec i incrémenté à chaque passage de la boucle           */
   printf("Saisissez la valeur de U%d\n", iIndiceElCour);

   /* Saisie de la valeur du ième entier */
   scanf("%d",&iTableau[iIndiceElCour]);
 }

/* Pour tous les éléments de la liste */
for(iIndiceElCour=0; iIndiceElCour<TAILLE; iIndiceElCour++)
 {
   /* Initialisation de l'élément de valeur minimum */
   iMin = iTableau[iIndiceElCour];

   /* Pour tous les éléments allant de l'élément courant */
   /* au dernier élément de la liste */
   for(iCompteur=iIndiceElCour+1; iCompteur<TAILLE; iCompteur++)
    {
      /* Si la valeur de l'élément courant n'est pas la valeur minimum */
      if (iMin > iTableau[iCompteur])
       {
        /* Alors, on échange l'élément courant avec celui de valeur minimum */
          iEchange = iTableau[iIndiceElCour];
          iTableau[iIndiceElCour] = iTableau[iCompteur];
          iTableau[iCompteur] = iEchange;

          /* Mise à jour du minimum */
          iMin = iTableau[iIndiceElCour];
       }
    }
  }

/*Affichage de la liste */
printf("La liste après tri est la suivante : \n");
for(iIndiceElCour=0; iIndiceElCour<(TAILLE-1); iIndiceElCour++)
  printf("%d, ",iTableau[iIndiceElCour]);
printf("%d\n ",iTableau[TAILLE-1]);
}
```

TRI A BULLE D'UN TABLEAU D'ENTIERS

```
#include<stdio.h>
/* Programme qui trie les éléments d'une liste d'entiers  (tri à bulle) */

int main()
{

/* DECLARATION DES VARIABLES */

int *iTableau=NULL;     /* Liste des entiers                                  */
int iNb_elements ;      /* Nombre d'éléments dans la liste                    */
int iCompteurDebut;     /* Compteur pour la boucle allant du début à la fin   */
int iCompteurFin;       /* Compteur pour la boucle allant de droite à gauche  */
int iEchange;           /* Variable temporaire pour l'échange de deux éléments */

/* CORPS DU PROGRAMME */

/* Saisie des éléments de la liste */
printf("Nombre d'éléments dans la liste :");
scanf("%d", &iNb_elements);
iTableau=(int*)malloc(iNb_elements*sizeof(int));

for(iCompteurDebut=0; iCompteurDebut<iNb_elements; iCompteurDebut++)
 {
   /* Affichage d'un message demandant la valeur du ième entier */
   /* avec i incrementé à chaque passage de la boucle           */
   printf("Saisissez la valeur de U%d\n",iCompteurDebut);

   /* Saisie de la valeur du ième entier */
   scanf("%d",&iTableau[iCompteurDebut]);
 }

/* Pour tous les éléments de la liste de gauche à droite */
for(iCompteurDebut =0;iCompteurDebut <iNb_elements;iCompteurDebut ++)
 {
   /* Pour tous les éléments à partir du dernier jusqu'au dernier élément trié */
  for(iCompteurFin =iNb_elements-1; iCompteurFin >iCompteurDebut; iCompteurFin --)
   {
     /* Si l'élément courant est plus petit que l'élément précédent */
     if (iTableau[iCompteurFin] < iTableau[iCompteurFin-1])
     /* ou if (iTableau[iCompteurFin] > iTableau[iCompteurFin-1])
        si on veut l'ordre décroissant */
      {
        /* Alors, on échange les éléments */
        iEchange = iTableau[iCompteurFin];
        iTableau[iCompteurFin] = iTableau[iCompteurFin-1];
        iTableau[iCompteurFin-1] = iEchange;
      }
    }
 }

/*Affichage de la liste */
printf("La liste après tri est la suivante : \n");
for(iCompteurDebut =0; iCompteurDebut <iNb_elements-1; iCompteurDebut ++)
 {
        printf("%d, ",iTableau[iCompteurDebut]);
 }
printf("%d\n ",iTableau[iNb_elements-1]);
free(iTableau);
}
```

TRI RAPIDE D'UN TABLEAU D'ENTIERS

```
/*                      FICHIER CONTENANT DES FONCTIONS PERMETTANT DE      */
/*                         TRIER RAPIDEMENT UN TABLEAU D'ENTIERS           */
/*                                                                         */
/* EXEMPLE : tableau = (3 7 1 4 2)                                         */
/*           on prend le premier élément comme pivot                       */
/*           Inf reçoit les éléments inférieurs au pivot : Inf = (1 2)     */
/*           Sup reçoit les éléments supérieurs au pivot : Sup = (7 4)     */
/*           le tableau concaténation de Inf, 3 (le pivot) et Sup avec     */
/*           Inf et Sup triés de la même manière (appel récursif)          */
/*                                                                         */

#include<stdio.h>

/* Fonction d'AFFICHAGE d'un tableau d'entiers : affiche */
/* Parametres : un tableau d'entiers : int* ipTableau    */
/*              la taille du tableau : int iTaille       */
/* Creation :  MM 13/09/2000                             */
/* Derniere mise a jour : MM  13/09/2000                 */
void affiche(int *ipTableau, int iTaille)
{
  int iCompteur;

  /* Affichage du tableau */
  printf("(");
  if(iTaille == 0) printf(")\n");
  else
    if(iTaille == 1) printf("%d)\n",*(ipTableau));
    else
       for(iCompteur=0;iCompteur<iTaille;iCompteur++)
       {
          if(iCompteur < iTaille -1) printf("%d ",*(ipTableau+iCompteur));
          else printf("%d)\n",*(ipTableau+iCompteur));
        }
}

/* Fonction de TRI RAPIDE d'un tableau d'entier : tri_rapide            */
/* Description : un pivot est choisi dans le tableau (le 1er élément)   */
/*               le tableau est découpé en deux sous-tableau Inf et Sup */
/*               contenant respectivement les entiers inférieurs et     */
/*               les entiers supérieurs au pivot.                       */
/*               le tableau renvoyé contient la concaténation de Inf    */
/*               trié (par appel récursif), du pivot et de Sup trié.    */
/*                                                                      */
/* Paramètre d'entrée : un tableau d'entiers : int *iPtableau           */
/*                      la taille de ce tableau : int iTaille           */
/* Retour : un tableau d'entiers (int *)                                */
/* Creation : MM 13/09/2000                                             */
/* Derniere mise a jour : 13/09/2000                                    */
int* tri_rapide(int * ipTableau, int iTaille)
{
  /* Pivot du tableau : 1er element */
  int iPivot;
  /* Tableau contenant les éléments < au pivot */
  int* ipInf;
  /* Nombre d'elements du tableau Inf */
  int iTailleInf=0;
  /* Tableau contenant les éléments > au pivot */
  int* ipSup;
  /* Nombre d'elements du tableau Sup */
  int iTailleSup=0;
  /* Compteur */
  int iCompteur;
```

```
  /* Si le tableau contient moins de 2 éléments, on ne fait rien */
  if (iTaille < 2) return ipTableau;
  else
    {
      iPivot=*ipTableau;

      /* Ecriture des tableaux Inf et Sup                 */
      /* Chacun contiendra au maximum iTaille elements -1 */
      ipInf=(int*)malloc(sizeof(int)*(iTaille-1));
      ipSup=(int*)malloc(sizeof(int)*(iTaille-1));

      for(iCompteur=1;iCompteur<iTaille;iCompteur++)
      {
        if(*(ipTableau+iCompteur)> iPivot)
          {
            *(ipSup+iTailleSup)=*(ipTableau+iCompteur);
            iTailleSup++;
          }
        else
          {
            *(ipInf+iTailleInf)=*(ipTableau+iCompteur);
            iTailleInf++;
          }
      } /* Fin du for */

      /* Reecriture du tableau à partir de Inf et Sup triés et du pivot */
      ipInf=tri_rapide(ipInf,iTailleInf);
      ipSup=tri_rapide(ipSup,iTailleSup);

      iTaille=0;

      /* Insertion des elements de Inf dans le tableau final */
      for(iCompteur=0;iCompteur<iTailleInf;iCompteur++)
      {
        *(ipTableau+iCompteur)=*(ipInf+iCompteur);
        iTaille++;
      }

      /* Insertion du pivot dans le tableau final */
      *(ipTableau+iCompteur)=iPivot;
      iTaille++;

      /* Insertion des elements de Sup dans le tableau final */
      for(iCompteur=0;iCompteur<iTailleSup;iCompteur++)
      {
        *(ipTableau+iTaille)=*(ipSup+iCompteur);
        iTaille++;
      }

      free(ipInf);
      free(ipSup);

      return ipTableau;

    }
}
```

```
/* PROGRAMME PRINCIPAL avec passage en paramètre d'un fichier           */
/* contenant en premier la taille d'un tableau puis les entiers du      */
/* tableaux séparés par des espaces                                     */
/* PAR EXEMPLE si le fichier contient la ligne : 9 10 5 12 2 9 1 3 4 7  */
/* Le resultat sera :                                                   */
/*                 > a.out
                   > Le tableau avant le tri est : (10 5 12 2 9 1 3 4 7)
                   > Le tableau après le tri est : (1 2 3 4 5 7 9 10 12) */
int main(int argc, char* argv[])
{
  /* Fichier contenant le tableau */
  FILE *FTableau;
  /* Taille du tableau */
  int iTaille;
  /* Tableau d'entiers */
  int *ipTableau;
  /* Compteur */
  int iCompteur;

  /* Si le nom du fichier n'est pas passé en paramètre, on sort */
  if(argc !=2) return 0;

  if((FTableau=fopen(argv[1],"r"))==NULL)
    {
      printf("Erreur Ouverture du fichier\n");
    }
  else
    { /* Lecture de la taille du tableau */
      fscanf(FTableau,"%d",&iTaille);


      /* Allocation mémoire du tableau d'entiers */
      ipTableau=(int*)malloc(iTaille*sizeof(int));

      /* Ecriture du tableau à partir des données du fichier */
      for(iCompteur=0;iCompteur<iTaille;iCompteur++)
      fscanf(FTableau,"%d",(ipTableau+iCompteur));

      printf("Le tableau avant le tri est : ");
      affiche(ipTableau,iTaille);

      ipTableau=tri_rapide(ipTableau,iTaille);

      printf("Le tableau après le tri est : ");
      affiche(ipTableau,iTaille);

      free(ipTableau);
      fclose(FTableau);
    }
}
```

UTILISATION DES FONCTIONS FREAD ET FWRITE

```
/*     PROGRAMME EXEMPLE D'UTILISATION DES FONCTIONS */
/*                 fread et fwrite                   */
/*                                                   */
/* Creation : MM 13/09/2000                          */

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

int main(int argc, char* argv[])
{
  /* Descripteur de fichier */
  FILE *Fichier;
  /* Chaîne de caractères utilisée pour afficher le contenu du fichier */
  char *cpChaine;

  /* le programme prend en paramètres deux arguments, le nom du fichier et */
  /* une chaîne de caractères sans espace                                   */
  if(argc <3)
      printf("usage : programme nom_fichier chaines_de_caracteres_sans_espace\n");

  else
    {

      if((Fichier=fopen(argv[1],"w+"))==NULL)
       printf("Erreur ouverture fichier\n");

      else
       {
       /* Ecriture de la chaine de cara passees en parametres dans le fichier */
       fwrite(argv[2],strlen(argv[2])+1,1,Fichier);

       /* Positionnement au debut du fichier */
       fseek(Fichier,SEEK_SET,0);

       cpChaine=(char*)malloc(strlen(argv[2]+1));

       /* Lecture et affichage du contenu du fichier */
       if(fread(cpChaine,strlen(argv[2])+1,1,Fichier)>0)
         printf("Contenu du fichier : %s\n",cpChaine);

       fclose(Fichier);

       }
    }
}
```