COMPTE-RENDU TP 7/8

SAAD BOUHYA

LES FONCTIONS PRINCIPALES:

On remplace le T_EMP par sa nouvelle structure.

rechercher un livre

```
int rechercherLivre(T_Bibliotheque  *ptrB, char
titre[MAX]){
 int compteur = 0;
 for(int i=0;i<ptrB->nbLivres;i++){

     if(strcmp( ptrB->etagere[i].titre , titre )
== 0 ){
       afficherLivre( &(ptrB->etagere[i])  );
       compteur++;
     }
 }
 return compteur;
}
```

rechercher un livre un livre par le nom d'auteur

```
int rechercherLivreAuteur(T_Bibliotheque  *ptrB,
char auteur[MAX]){
   int compteur = 0;

   for(int i=0;i<ptrB->nbLivres;i++){
     if(strcmp( ptrB->etagere[i].auteur , auteur )
== 0 ){
       afficherLivre( &(ptrB->etagere[i])  );
       compteur++;
     }
   }
   return compteur;
}
```

Supprimer un livre:

```
int supprimerLivre(T_Bibliotheque  *ptrB, char
titre[MAX]){
 int i, compteur = 0;
  for(i=0 ;i<ptrB->nbLivres;i++){

       if(strcmp( ptrB->etagere[i].titre , titre )
== 0 ){
         compteur++;
         break;
       }
   }
```

```
 for(i; i<ptrB->nbLivres; i++){
   memcpy(&(ptrB->etagere[i]),
&(ptrB->etagere[i+1]), sizeof(T_livre));
 }
 (ptrB->nbLivres)--;
 return compteur;

}
```

Emprunter un livre:

```
int emprunterLivre(T_Bibliotheque  *ptrB, char
nom_emprunteur[MAX], int position) {

strcpy((ptrB->etagere[position].emprunteur.nomempr
unteur), nom_emprunteur);
}
```

Rechercher un livre par code

```
int rechercherCode(T_Bibliotheque  *ptrB, char
code[MAX_CODE]){
 for(int i=0;i<ptrB->nbLivres;i++){
     if(strcmp( ptrB->etagere[i].code , code ) ==
0 ){
       return i;
     }
 }
```

```
 return -1;
}
```

Rendre un livre:

```
void rendreLivre(T_Bibliotheque  *ptrB, int
position) {

strcpy((ptrB->etagere[position].emprunteur.nomempr
unteur), "");
}
```

Trier les livres par titre du livre:

```
void trierParTitre(T_Bibliotheque  *ptrB) {
 T_tabloDeLivres aux;
 for(int i=0; i<ptrB->nbLivres;i++){
     for(int j = 1; j < (ptrB->nbLivres)-i; j++){

if(strcmp((ptrB->etagere[j-1].titre),(ptrB->etager
e[j].titre)) > 0){
          aux[0] = (ptrB->etagere[j-1]);
          (ptrB->etagere[j-1]) =
(ptrB->etagere[j]);
          (ptrB->etagere[j]) = aux[0];
          // strcpy(aux,
(ptrB->etagere[j-1].titre));
```

```
        // strcpy((ptrB->etagere[j-1].titre),
(ptrB->etagere[j].titre));
        // strcpy((ptrB->etagere[j].titre), aux);
      }
    }
 }
}
```

Trier les livres par nom d'auteur:

```
void trierParAuteur(T_Bibliotheque  *ptrB){
 T_tabloDeLivres aux;
 for(int i=0; i<ptrB->nbLivres;i++){
     for(int j = 1; j < (ptrB->nbLivres)-i; j++){

if(strcmp((ptrB->etagere[j-1].auteur),(ptrB->etage
re[j].auteur)) > 0){
         aux[0] = (ptrB->etagere[j-1]);
         (ptrB->etagere[j-1]) =
(ptrB->etagere[j]);
         (ptrB->etagere[j]) = aux[0];
      }
    }
 }
}
```

Trier par annee:

```
void trierParAnnee(T_Bibliotheque  *ptrB){
 T_tabloDeLivres aux;
 for(int i=0; i<ptrB->nbLivres;i++){
     for(int j = 1; j < (ptrB->nbLivres)-i; j++){
       if((ptrB->etagere[j-1].annee) >
(ptrB->etagere[j].annee)) {
         aux[0] = (ptrB->etagere[j-1]);
         (ptrB->etagere[j-1]) =
(ptrB->etagere[j]);
         (ptrB->etagere[j]) = aux[0];
       }
     }
 }
}
```

Pour afficher les livres disponibles:

```
int livre_disponible(T_Bibliotheque *ptrB) {
 // if
(strcmp((ptrB->etagere[position].emprunteur), "")
== 0)
 //  return 1;
 // else
 //  return 0;
 int compteur;
 for(int i=0;i<ptrB->nbLivres;i++) {
```

```
   if
(strcmp((ptrB->etagere[i].emprunteur.nomemprunteur
), "") == 0) {
     printf("CODE : %s\n", ptrB->etagere[i].code);
     compteur++;
   }
 }
 if(compteur == 0)
   printf("y a pas de livre dispo\n");
}
```

Pour effectuer le chargement des livres:

```
void chargement(T_Bibliotheque *ptrB)
{
FILE *fic=NULL; //le type FILE
int i=0;
fic=fopen("baseLivres","r"); // r = le mode read
//fopen renvoie NULL si probleme (disque plein,
disque non accessible ...
if (fic!=NULL)
 {
 do
   {

   fread(  &(ptrB->etagere[i])
,sizeof(T_livre),1,fic);
   i++;
```

```
   }
   while(!feof(fic));
 fclose(fic);
 ptrB->nbLivres=i-1;
 puts("CHARGEMENT  REUSSI ..............");
 }
 else puts("ECHEC DE CHARGEMENT  !!!!!  ");

}
```

Pour effectuer la sauvegarde des donnees:

```
void sauvegarde(T_Bibliotheque *ptrB)
{
FILE *fic=NULL; //le type FILE
int i;
fic=fopen("baseLivres","w"); // w = le mode =
write avec ECRASEMENT
//fopen renvoie NULL si probleme (disque plein,
disque non accessible ...
if (fic!=NULL)
 {
 for(i=0;i<ptrB->nbLivres;i++)
   {
//fwrite(const void *ptr, size_t size, size_t
nmemb, FILE *stream)
   fwrite(  &(ptrB->etagere[i])
,sizeof(T_livre),1,fic);
```

```
   }
 fclose(fic);
 puts("SAUVEGARDE REUSSIE ..............");



 }
 else puts("ECHEC DE SAUVEGARDE  !!!!!  ");

}
```

Les livres disponibles:

```
int livre_disponible(T_Bibliotheque *ptrB) {
 // if
(strcmp((ptrB->etagere[position].emprunteur), "")
== 0)
 //  return 1;
 // else
 //  return 0;
 int compteur;
 for(int i=0;i<ptrB->nbLivres;i++) {
   if
(strcmp((ptrB->etagere[i].emprunteur.nomemprunteur
), "") == 0) {
     printf("CODE : %s\n", ptrB->etagere[i].code);
     compteur++;
```

```
   }
 }
 if(compteur == 0)
   printf("y a pas de livre dispo\n");
}
```

Les emprunts en retartd:

```
void emprunts_retard (T_Bibliotheque *ptrB) {
 int i;
 int tab[12] = {0, 31, 59, 90, 120, 151, 181, 212,
243, 273, 304, 334};
 int valeur;
 int calcul;
 int resultat;
  for (i=0; i<ptrB->nbLivres; i++) {

if(strcmp(ptrB->etagere[i].emprunteur.nomemprunteu
r, "") != 0) {
     //lireDateSysteme(
&(ptrB->etagere[i].emprunteur) );
     calcul = (365 *
(ptrB->etagere[i].emprunteur.lannee) +
tab[(ptrB->etagere[i].emprunteur.lemois)] +
(ptrB->etagere[i].emprunteur.ladate));
     valeur = lireDateSysteme(
&(ptrB->etagere[i].emprunteur) );
     resultat = valeur - calcul;
```

```
      if (resultat > 7) {
        printf("le livre '%s' est en retard\n",
ptrB->etagere[i].titre);
      }
    }
  }
}
```