SDD : TP 1

March 11, 2018

Présentation générale

- Ce TP à pour but de créer une matrice représentant des coûts de production de différentes usines à différentes périodes depuis un fichier. Il faut ensuite récupérer les K usines ayant le plus faible coût de production. Finalement, il faut toutes les occurences d'une usines u dans cette liste et les supprimer.

- schéma

- Les fichiers sources se trouvent dans le dossier **src**. Les entêtes sont dans le dossier **include**.

1 Détail de chaque fonction

1.1 findElmt

Principe : FindElmt

On initialise un pointeur courant qui va parcourir la liste.
Tant qu'il n'est pas arrivé à la fin et que l'élément est inférieur à v
    On parcout la liste chainée.
On retourne l'adresse du pointeur courant
FIN

Lexique :

- Paramètre(s) de la fonction
  - pHead est la tête fictive de la liste chainée
  - v est la valeur que l'on cherche dans cette liste.
- Variable(s) locale(s)
  - curr est le pointeur courant qui parcourt la liste.

1.2 InserKSorted

Principe : FindElmt

On initialise un pointeur courant qui va parcourir la liste.
Tant qu'il n'est pas arrivé à la fin et que l'élément est inférieur à v
    On parcout la liste chainée.
On retourne l'adresse du pointeur courant

Lexique :

- Paramètre(s) de la fonction
  - Phead est la tête fictive de la liste chainée
  - v est la valeur que l'on cherche dans cette liste.
- Variable(s) locale(s)
  - curr est le pointeur courant qui parcourt la liste.

1.3 RemoveFactory

Principe : RemoveFactory

On initialise un pointeur courant, un pointeur précédent et un pointeur temporaire.
Tant qu'on est pas arrivé à la fin de la liste.
    Si l'entreprise courante est celle recherchée alors :
        On supprime cet élément de la liste.
    Sinon
        On avance dans la liste.

FIN

Lexique :

- Paramètre(s) de la fonction
  - pHead est la tête fictive de la liste chainée
  - factory est l'indice de l'usine que l'on supprimé de la liste chainée.
- Variable(s) locale(s)
  - prev est le pointeur qui pointe vers l'élément précédant le pointeur curr.
  - curr est le pointeur courant qui parcourt la liste.
  - tmp est un pointeur qui permet de libérer un bloc proprement.

1.4 WriteLinkedListToFile

Principe : WriteLinkedListToFile

On initialise un pointeur courant
Tant qu'on est pas arrivé à la fin de la liste
    On affiche un message avec des données de l'élément courant
    Le pointeur courant avance dans la liste

FIN

Lexique :

- Paramètre(s) de la fonction
  - file est le fichier dans lequel on veut afficher la liste chainée. (stdout, stderr, ou un fichier texte).
  - pHead est la tête fictive de la liste chainée.
- Variable(s) locale(s)
  - curr est le pointeur courant qui parcourt la liste.

1.5 InsertProdutionBlock

Principe : InsertProdutionBlock

On récupère l'adresse à laquelle on doit insérer le nouvel élément afin de garder la liste triée.
On crée un nouveau bloc.
Si on est capable d'allouer le bloc alors
    On affecte les nouvelles valeurs à ce bloc.
    On l'insert dans la liste chaînée triée.
Sinon
    On affiche un message d'erreur

FIN

Lexique :

- Paramètre(s) de la fonction
  - pHead est la tête fictive de la liste chainée.
  - value est le coût de production de l'usine.
  - factory est l'indice de l'usine.
  - period est la période de production de l'usine.
  - K est le nombre de plus petit coût de production que l'on veut garder.
- Variable(s) locale(s)
  - insertAdress pointe vers le bloc où on doit insérer le nouveau bloc.
  - newElement pointe vers le nouveau bloc.

1.6 freeLinkedList

Principe : freeLinkedList

Tant qu'on est pas arrivé au bout de la liste faire
    On libère proprement le bloc courant.
    On passe à l'élément suivant.
FIN

Lexique :

- Paramètre(s) de la fonction
  - pHead est la tête fictive de la liste chainée.
- Variable(s) locale(s)
  - curr est le pointeur courant qui parcourt la liste.
  - tmp est un pointeur qui permet de libérer un bloc proprement.

1.7 loadMatrixFromFile

Principe : loadMatrixFromFile

On crée un flux vers notre fichier contenant notre matrice et une matrice pour stocker ses valeurs, sa ligne, et sa colonne.

Si le fichier s'est ouvert correctement alors
    On lit les dimensions de la matrice situées sur la première ligne du fichier
    On alloue dynamiquement une matrice.
    Si on a un problème d'allocation à un certain moment
        on libère proprement la matrice et on rapporte une erreur à la fonction appelante.
    Sinon
        On lit l'élément $m_{i,j}$ depuis le fichier et on l'insère dans la matrice
    On ferme le fichier
Sinon
    On rapporte une erreur à la fonction appelante.
FIN

Lexique :

- Paramètre(s) de la fonction
  - fileName est le nom du fichier qui contient la matrice.
  - errorCode est un pointeur sur un entier qui indique si la fonction s'est bien passée.
- Variable(s) locale(s)
  - matrix est une structure contenant les valeurs de la matrice et sa taille.
  - issue = 1, signifie qu'on doit s'arrêter d'allouer la matrice.
  - file est le flux que l'on ouvre avec le nom du fichier contenant la matrice
  - i et j permettent de parcourir la matrice.

1.8 freeMatrix

Principe : freeMatrix

Si la matrice est non nulle alors :
    On parcourt les lignes de la matrice :
        Si la valeur du bloc de la ligne i est non nulle alors:
            On libère ce bloc
    On libère la matrice;
FIN

Lexique :

- Paramètre(s) de la fonction
  - matrix est la structure de la matrice que l'on veut afficher
- Variable(s) locale(s)
  - i est l'entier qui permet de supprimer chaque ligne de la matrice.

1.9 printMatrix

Principe : printMatrix

On parcourt les lignes de la matrice
On parcourt les colonnes de la matrice
On affiche l'élément de la i-ème ligne et de la j-ème colonne

FIN

Lexique :

- Paramètre(s) de la fonction
  - matrix est la structure de la matrice que l'on veut afficher.
- Variable(s) locale(s)
  - i et j permettent d'afficher l'élément de la i-ème ligne et de la j-ème colonne.