## Consignes générales pour les TDs

FAITES BIEN LES SCHEMAS DEMANDES, **CE N’EST PAS OPTIONNEL** ET VOUS AIDERA À VISUALISER ET COMPRENDRE LES ALGORITHMES

POUR LES EXERCICES COMPORTANT DE LA REDACTION DE CODE EN C, **UTILISEZ UN EDITEUR DE TEXTE OU UN IDE**, AFIN DE POUVOIR RECUPERER VOS TRAVAUX POUR LES TPs.

Les exercices précédés d’un symbole losange ◆ sont obligatoires.

# TD2 : Listes Avancées

### Thème 1 - Listes ‘ht’ avec pointeurs vers tête et fin de liste – objectif : visualiser les structures de données – concevoir et rédiger les fonctions classiques de traitement

On rappelle la structure utilisée pour ces listes :

**typedef struct s\_ht\_list**

**{**

**t\_cell \*head;**

**t\_cell \*tail;**

**} t\_ht\_list;**

* *Exercice 1)*

Écrivez la fonction **t\_ht\_list createHtList()** qui retourne une list ht vide

* *Exercice 2)*

Question 1) Quelle condition doit-on tester pour vérifier qu’une t\_ht\_list est bien vide ?

Question 2) On ajoute une nouvelle cellule pointée par un pointeur **newcell**, stockant la valeur -4, à une t\_ht\_list vide, nommée **myhtlist**. Faites la visualisation de l’état de **myhtlist** après cet ajout.

Question 3) Écrivez le programme qui permet de faire cet ajout (vous devrez appeler la fonction **createCell()** pour créer la cellule à ajouter. On vous en donne les premières lignes :

**t\_ht\_list myhtlist ;**

**t\_cell \*newcell ;**

**myhtlist = createHtList() ;**

**// à vous de continuer**

Question 4) On veut ajouter une nouvelle cellule pointée par un pointeur **newcell**, stockant la valeur 123, en tête de **myhtlist**. Faites la visualisation de l’état de **myhtlist** après cet ajout en tête.

Question 5) Quelle est la différence avec le premier ajout en tête ?

Question 6) Que remarquez-vous pour le pointeur **tail** ?

Question 7) Écrivez le programme qui permet de faire cet ajout (vous devrez appeler la fonction **createCell()** pour créer la cellule à ajouter)

Suite du programme précédent :

Question 8) On souhaite maintenant ajouter une cellule **en fin de liste**, avec la valeur 7. Faites la visualisation de l’état de **myhtlist** après cet ajout en tête.

Question 9) Écrivez le programme qui permet de faire cet ajout (vous devrez appeler la fonction **createCell()** pour créer la cellule à ajouter)

Suite du programme précédent :

Question 10) Écrivez une fonction **AddTailHt()** qui ajoute, en fin de liste, une cellule dont la valeur est fournie en paramètre. N’oubliez pas de vous poser la question de savoir si la liste peut être modifiée par cette fonction. (vous devrez appeler la fonction **createCell()** pour créer la cellule à ajouter)

### Thème 2 : listes circulaires

* *Exercice 3)* Faites la visualisation d’une liste circulaire stockant les 3 valeurs 4, 7 et 1, dans cet ordre
* *Exercice 4)* Pourquoi dont-on traiter différemment l’insertion en tête de cellule dans une liste circulaire vide et dans une liste circulaire non-vide ?
* *Exercice 5)*

Écrivez la fonction **displayCircList()** qui affiche les éléments d’une liste circulaire, du premier au dernier, en suivant les questions :

Question 1) Rappelez la condition pour tester qu’une liste circulaire est vide.

Question 2) écrivez le corps de la fonction en vous basant sur la base suivante, code C en bleu, algorithme en vert (page suivante)

**void displayCircList(t\_circ\_list clist)**

**{**

**t\_cell \*temp = clist.head ;**

**si la liste n’est pas vide alors**

**{**

**Tant que temp ne pointe pas sur la dernière cellule**

**{**

**Afficher la valeur de la cellule pointée par temp**

**Passer à la cellule suivante**

**}**

**Afficher la valeur de la dernière cellule**

**}**

**return;**

**}**

Question 3) On s’intéresse à la forme particulière de ce programme. Pourquoi faut-il ne pas traiter la dernière cellule lors du parcours et l’afficher ensuite ? pour cela, posons-nous plusieurs questions

3.1) Supposons que l’on décide de traiter cette dernière cellule dans la boucle. On s’arrêterait donc de retour à la première. Comment écrirait-on la boucle **while** dans ce cas ?

3.2) que vaut **temp** avant le premier test de la condition de la boucle **while** ?

4.3) qu’en concluez-vous ?

*Exercice 6)* implémentation de la fonction de suppression de cellule vue en cours

Cet exercice peut paraître long, mais il est décomposé en de très nombreuses étapes

On souhaite écrire le corps de la fonction **removeCellCircList()** , dont on vous donne le prototype :

**void removeCellCircList(t\_circ\_list \*p\_list, int val) ;**

Le rôle de cette fonction est de supprimer, si elle existe, la première occurrence de la cellule stockant la valeur val (deuxième paramètre), dans une liste dont on à l’adresse (pointeur) en premier paramètre.

Question 1) pourquoi doit-on fournir l’adresse de la liste à traiter ?

Question 2) Rappelez la condition pour tester qu’une liste circulaire ne contienne qu’une cellule

Question 3) Vous trouverez en page suivante le squelette de cette fonction : code C en bleu, algorithme en vert – prenez bien le temps de le lire – vous en retrouverez les explications dans le support de cours. Écrivez les instructions correspondant à l’algorithme proposé.

**void removeCellCircList(t\_circ\_list \*p\_list, int val)**

**{**

**p\_cell temp, prev ;**

**Si la liste n’est pas vide**

**{**

**Si la liste ne contient qu’une seule cellule et que la cellule stocke la valeur**

**{**

**Reinitialiser la liste**

**}**

**Sinon**

**{ // on cherche la cellule**

**Positionner temp et prev au début de la liste**

**Tant que (on n’est pas à la fin de la liste et que temp ne stocke pas la valeur cherchée)**

**{**

**Sauvegarder temp dans prev**

**temp passe à la cellule suivante**

**}**

**Si on n’est pas à la fin de la liste // on a trouve la cellule (sauf si c’est la dernière) comme pour l’affichage**

**{**

**Si c’est la première**

**{**

**La nouvelle première est la suivante de l'actuelle première que l'on veut supprimer**

**La dernière pointe sur cette nouvelle première**

**}**

**Sinon**

**{**

**La suivante de prev devient la suivante de temp**

**}**

**}**

**// on vérifie si c’est la dernière**

**Si la dernière stocke la valeur cherchée**

**{**

**// temp et prev sont bien placés !**

**La suivante de prev devient la suivante de temp**

**la nouvelle dernière est prev**

**}**

**}**

**}**

**return;**

**}**

### Thème 3 - Listes doublement chaînées

*Exercice 7)*

A partir des slides 37 et 38 du CM #2, nous allons écrire la fonction **addHeadDoubleList(),** qui ajoute en tête de liste double une cellule dont la valeur entière est fournie en paramètre

Question 1) Cette fonction modifie-t-elle la liste (au sens vu lors du premier cours) ?

Question 2) les instructions proposées fonctionnent-elles avec une liste vide ?

Question 3) D’après la réponse à la question précédente, écrivez la fonction **addHeadDoubleList()** . Vous utiliserez la fonction **createDCell()** pour créer la nouvelle cellule à ajouter en tête de liste.

*Exercice 8)* Suppression et insertion en tête et en queue de liste double

Question 1) faites un schéma indiquant comment réaliser une suppression en tête d’une **liste non vide** pointée par un pointeur **ptr\_dlist** (conseil : dessinez les variables en noir et les ‘flèches’ (pointeurs à modifier) dans une autre couleur. Écrivez ensuite les instructions correspondantes

Question 2) faites un schéma indiquant comment réaliser une suppression en queue d’une liste non vide pointée par un pointeur **temp** , c’est-à-dire que **temp** pointe sur la dernière cellule, à supprimer

(Conseil : dessinez les variables en noir et les ‘flèches’ (pointeurs à modifier) dans une autre couleur. Écrivez ensuite les instructions correspondantes

Question 3) Faites un schéma indiquant comment réaliser une insertion en tête d’une liste **vide** pointée par un pointeur **ptr\_dlist**, d’une cellule pointée par un pointeur **newdcell** (conseil : dessinez les variables en noir et les ‘flèches’ (pointeurs à modifier) dans une autre couleur. Écrivez ensuite les instructions correspondantes

Question 4) insertion en tête de liste non vide : en fait, nous avons déjà traité ce cas. À quoi cetteLopération est-elle équivalente ?

Pour démarrer : faites un schéma indiquant comment réaliser une insertion en tête d’une liste **non** **vide** pointée par un pointeur **ptr\_dlist**, d’une cellule pointée par un pointeur **newdcell** (conseil : dessinez les variables en noir et les ‘flèches’ (pointeurs à modifier) dans une autre couleur. Écrivez ensuite les instructions correspondantes.