## 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.

# TD3 : Piles et Files

Thème 1 – Fonctions de la pile

* Exercice 1 ) dépiler

Retour à la fonction **unstack()**(Slide 41 de la présentation du cours)…

Le but de cet exercice est d’écrire la fonction **unstack()**de façon à ce qu'elle puisse gérer tous les cas qui peuvent se présenter...

Soit **mystack** une pile, stockant des valeurs entières **int.**

Pour le moment, savoir si la pile est implémentée en utilisant une liste ou un tableau n'est pas important: nous utiliserons le type 'générique' **t\_stack** pour représenter une pile.

Question 1) Pour l'instant, le prototype de **unstack()** est : **int unstack(t\_stack \*);**

1.1) Expliquez pourquoi le type de paramètre est **t\_stack \*.**

1.2) À quoi la valeur de retour est-elle censée servir ?

1.3) Quelle est la condition à tester avant de pouvoir dépiler ?

Question 2) Question à choix multiple : Que se passe-t-il si la condition précédente n'est pas remplie ?

a) Le programme devrait se planter

b) Le programme doit s'arrêter et sortir

c) Le programme devrait afficher le message "cannot unstack() from an empty stack".

d) Le programme doit être capable de savoir que la fonction n'a pas pu dépiler une valeur.

Question 3) Si l'on essaie de dépiler à partir d'une pile vide, la valeur de retour de la fonction **unstack()** ne sera pas pertinente; donc **unstack()**ne doit être appelée que si la pile n'est pas vide.

Écrivez les définitions de la fonction générique **unstack()**, en supposant que la pile n'est pas vide :

3.1) écrivez la fonction **unstack\_tab()** pour une pile implémentée avec un tableau

3.2) écrire la fonction **unstack\_list()** pour une pile implémentée avec une liste

Question 4) En utilisant **isEmptyStdList()**, écrivez la fonction **isEmptyStackList()**

Remarque : la fonction **isEmptyStackTab()** est déjà disponible slide 36 du CM3.

Question 5) À partir de la définition du type **t\_stacktab** (slide 32 du CM3), écrivez la définition de la fonction **isFullStackTab()**

* Exercice 2) empiler()

Question 1) En utilisant une pile implémentée avec un tableau, quelle est la condition à vérifier avant d'essayer d'empiler un élément ?

Question 2) Nous allons écrire les définitions de la fonction générique **stack()**, en supposant que la pile n'est pas pleine :

2.1) Écrivez la fonction **stack\_tab()** pour une pile implémentée avec un tableau

2.2) écrire la fonction **stack\_list()** pour une pile implémentée avec une liste. Veuillez réutiliser autant que possible les fonctions déjà existantes.

Thème 2 – Fonctions sur les files

* Exercice 3) Files

Rappel : les valeurs stockées dans les files ne sont plus des valeurs entières (**int**) mais de type **t\_customer**.

Cela ne changera presque rien, sauf que les noms de type à utiliser pour les files que nous traiterons par la suite sont :

**t\_queue\_cust** pour les listes, **t\_queuetab\_cust** pour les tableaux

Question 1) Représenter / dessiner l'état d'une file, implémentée avec une liste, où : 3 éléments ont été mis en file d'attente, puis tous ont été retirés de la file d'attente.

Question 2) Écrivez la définition des fonctions suivantes pour les files:

**2.1) isEmptyQueueCust()** pour une file d'attente implémentée avec une liste

**2.2) isEmptyQueueTabCust()** pour une file d'attente implémentée avec un tableau

Question 3) Pour les files d'attente implémentées avec une **t\_ht\_list**, expliquez comment choisir entre :

a) Enqueue en tête de liste / dequeue en queue de liste

b) Enqueue en queue de liste / dequeue en tête de liste

Question 4) En supposant que les files d'attente ne sont pas vides, nous allons implémenter la fonction générique **dequeue()**. Rappelez-vous que les files d'attente stockent des valeurs de type **t\_customer**.

4.1) Écrire le prototype de la fonction **dequeueTabCust()**

4.2) Écrire le prototype de la fonction **dequeueListCust()**

4.3) Écrire la définition de la fonction **dequeueTabCust()**

4.4) Écrire la définition de la fonction **dequeueListCust()**

4.5) Enqueue avec une liste: écrire le prototype de **enqueueListCust()** : cette fonction ajoute un nouveau **t\_customer** à la file.

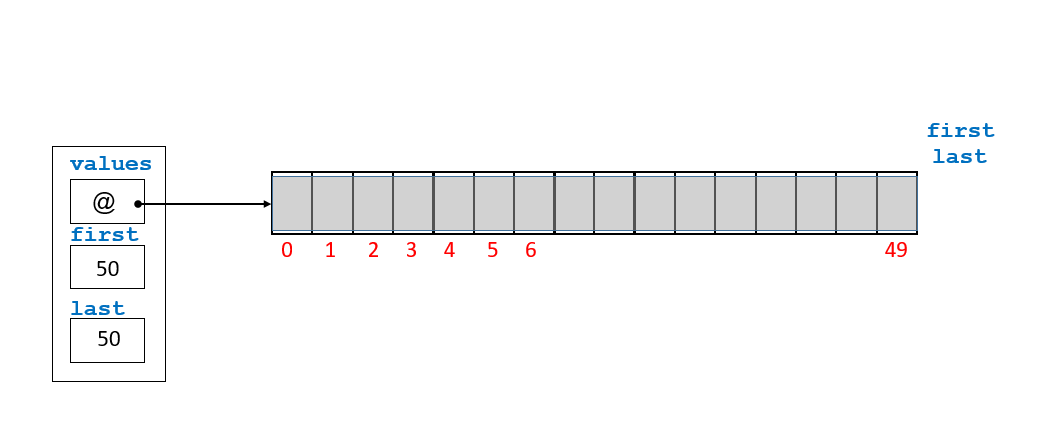
4.6) Enqueue avec une liste: écrire la définition de **enqueueListCust().** Pour créer une nouvelle cellule stockant un client, on admet que l'on peut appeler la fonction suivante : **t\_cell\_cust \*createCellCust(t\_customer) ;** qui joue le même rôle que la fonction **t\_cell \*createCell(int) ;** pour des valeurs entières.

4.7) Enqueue avec un tableau : écrire le prototype de **enqueueTabCust()** : cette fonction ajoute un nouveau **t\_customer** à la file.

4.8) Enqueue avec un tableau : écrire la définition de **enqueueTabCust()**

* Exercice 4) Files d'attente avec tableaux : gestion d'une file d'attente "pleine".

Rappel : la file d'attente est vide :



Question 1) Concrètement combien de valeurs devrions-nous être en mesure de stocker ?

Question 2) Idéalement, quel serait l'endroit suivant pour ajouter – **enqueue()** - une nouvelle valeur ?

Supposons que nous voulons enqueue une nouvelle valeur : enqueue consiste à placer la valeur dans le tableau à l'index **last**, puis à l'incrémenter **last.**

Question 3) Nous voudrions que **enqueue()** soit "cyclique", c'est-à-dire que lorsqu'on atteint la fin du tableau, on utilise à nouveau le début du tableau s'il reste des places disponibles.

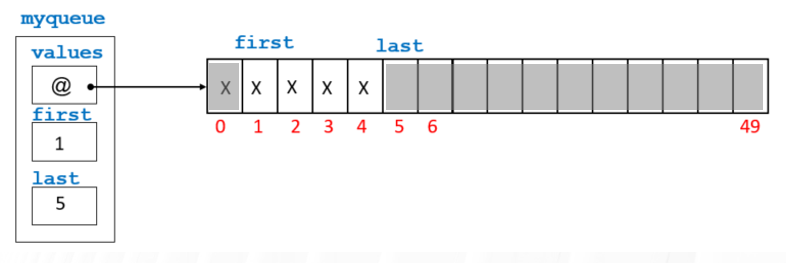
C’est-à-dire, nous voudrions que **last=50** indique l'indice : 0, **last=51** indique l'indice : 1, encore et encore...

Comment calculer un indice dans l'intervalle 0..49 à partir du **last**, qui s'incrémente sans limites ?

La même affirmation est également vraie pour **first.**

Question 4) Comment vérifier que la file d'attente est pleine ?

A partir du schéma suivant, indiquer le nombre de places utilisées dans la file, et trouver la relation entre ce nombre et les valeurs de **first** et **last**



En généralisant cette relation, quelle est la condition qui indique que la file d'attente est pleine ?

Exercice 5) Aller plus loin avec files implémentés par des tableaux (stockage des clients)

Nous voudrions définir une structure plus flexible pour gérer les files d'attente avec des tableaux, en utilisant des tableaux dynamiques au lieu de tableaux statiques. Il faut aussi stocker la taille du tableau.

Question 1) Quelle est la définition de type, en tant que struct, à utiliser ?

Question 2) Écrivez la définition de la fonction :

**t\_queuetab\_cust createQueueueTabCust(int);** où le paramètre est la taille du tableau à créer.

Consignes : la fonction doit vérifier que le paramètre est >=0. Si non, la taille devra être 0.