Programmation C – Tableaux dynamiques

Durée : 2h, aucun document autorisé
Épreuve papier uniquement (pas d'épreuve machine)

Préambule

Reporter clairement sur la copie le numéro de chaque question, même lorsqu'aucune réponse n'est fournie.

1 Introduction

En **C**, la taille d'un tableau ne peut pas changer au cours de l'exécution d'un programme. Cela est gênant lorsqu'on souhaite stocker une certaine quantité de données mais que l'on ne connaît pas à l'avance le nombre d'éléments à stocker. Dans cet examen, on se propose de programmer des fonctions associées à une structure **TableauDynamique** dont l'objectif est précisément de gérer le stockage d'un nombre non défini à l'avance d'entiers (de type **int**).
La structure **TableauDynamique** est définie comme suit :

```
struct TableauDynamique{
  int n_elements;
  int *memoire;
  int capacite;
};
```

Et le type pointeur **pTableauDyamique** associé est défini comme :

```
typedef struct TableauDynamique *pTableauDynamique;
```

Les champs de **TableauDynamique** possèdent la signification suivante :
– **n_elements :** nombre d'entiers stockés dans le tableau ;
– **memoire :** pointeur sur une zone de la mémoire destinée à recevoir **capacite** entiers de type **int**.
Evidemment, à toute étape du programme, **n_elements** est plus petit ou égal à **capacite**. Si jamais le nombre d'entiers **n_elements** à stocker dans le tableau dynamique devait devenir supérieur à **capacite**, alors une nouvelle allocation de mémoire est effectuée pour recevoir les nouveaux **int** à stocker : en d'autres termes, le tableau est redimensionné.

2 Travail à faire

Dans un fichier **TableauDyamique.c**, programmer les fonctions suivantes (pour simplifier les fonctions, on supposera que les paramètres de type **pTableauDynamique** passés aux fonctions sont toujours différents de **NULL**).

1. pTableauDynamique creeTableau().
   Cette fonction crée un pointeur sur une structure de type **TableauDynamique**, et c'est ce pointeur qui est renvoyé. Lors de la création d'un tableau, le nombre d'éléments stockés est évidemment égal à 0. En revanche, il est préférable d'allouer dès le départ de l'espace mémoire pour quelques entiers : on choisira ici d'allouer de la mémoire pour 10 **int** (i.e. **memoire** est un pointeur sur une zone suffisamment grande pour **capacite**=10 entiers de type **int**).
2. **void** detruitTableau (pTableauDynamique tableau).
   Cette fonction libère la mémoire utilisée par le tableau dynamique pointé par **tableau** (attention à libérer toute la mémoire nécessaire).

3. pTableauDynamique recopieTableau(pTableauDynamique tableau).
   Crée un nouveau tableau dynamique qui est la copie du tableau **tableau** passé en paramètre et renvoie l'adresse de ce tableau. (Indication : pour effectuer cette recopie, une nouvelle allocation mémoire de **tableau→capacite** entiers doit être effectuée.)
4. pTableauDynamique doubleCapacite(pTableauDynamique tableau).
   Double la capacité du tableau **tableau** passé en paramètre et renvoie un pointeur sur le tableau dont la capacité a été ajustée. L'algorithme est simple :
   (a) allocation de mémoire pour 2×**tableau→capacite** entiers de type **int** ;
   (b) recopie des **tableau→n_elements** éléments de **tableau→memoire** dans la zone mémoire allouée ;
   (c) libération de la zone mémoire adressée par **tableau→memoire** ;
   (d) ajustement de **tableau→memoire** pour que ce pointeur désigne la zone mémoire allouée au début de l'algorithme ;
   (e) **tableau→capacite = 2*tableau→capacite ;**
5. **void** ajoute(pTableauDynamique tableau, **int** i).
   Ajoute l'entier **i** à la fin du tableau correspondant au pointeur **tableau**. Si le nombre d'éléments dans le tableau ne permet pas d'ajouter simplement l'entier **i** à cause d'une capacité limitée, alors la capacité du tableau est préalablement doublée grâce à la fonction **doubleCapacite**. (Ne pas oublier d'ajuster **n_elements** de manière adéquate.)
6. **void** ajoute_position (pTableaDynamique tableau, **int** i, **int** position).
   Même chose que la fonction précédente sauf que l'entier **i** est inséré à la position **position** du tableau. Si **position** est strictement inférieure à 0, alors l'insertion se fait en tête de tableau ; si **position** est supérieure ou égale à **tableau→n_elements**, l'insertion se fait en fin de tableau. Il faudra bien évidemment prendre soin de décaler comme il faut les éléments dont la position se trouve après **position** dans le tableau.
7. **int** supprime(pTableauDynamique tableau, **int** position).
   Supprime du tableau pointé par **tableau** l'élément à la position **position** (mêmes règles que précédemment pour les dépassements d'indice concernant **position**). Renvoie la valeur de l'entier à la position **position** avant suppression. Un décalage des valeurs dans le tableau doit être effectué si nécessaire.
8. **int** indexOf(pTableauDynamique tableau, **int** i).
   Renvoie l'indice de la première occurrence de l'entier **i** dans le tableau pointé par **tableau**. Renvoie -1 si **i** n'est pas dans le tableau.
9. **int** lastIndexOf (pTableauDynamique tableau, **int** i).
   Renvoie l'indice de la dernière occurrence de l'entier **i** dans le tableau pointé par **tableau**. Renvoie -1 si **i** n'est pas dans le tableau.
10. **int** contains (pTableauDynamique tableau, **int** i).
    Renvoie 1 si le tableau pointé par **tableau** contient l'entier **i** et 0 sinon.
11. **int** equals (pTableauDynamique tableau1, pTableauDynamique tableau2).
    Renvoie 1 si les deux tableaux pointés par **tableau1** et **tableau2** contiennent les mêmes entiers aux mêmes positions (attention, les capacités des 2 tableaux peuvent être différentes) et 0 sinon.

Répondre à présent aux questions suivantes :

12. Écrire le fichier en-tête **TableauDynamique.h** correspondant au fichier **TableauDynamique.c**.
13. Écrire deux fichiers **Ecriture.h** et **Ecriture.c** qui fournissent une fonction
    void affiche (pTableauDynamique tableau)
    dédiée à l'affichage du tableau dynamique pointé par **tableau**.
14. En imaginant qu'il existe un programme principal enregistré dans un fichier **main.c** qui utilise à la fois des fonctions déclarées dans **TableauDynamique.h** et celle déclarée dans **Ecriture.h**, écrire le fichier **makefile** permettant de créer un exécutable.