Allocation dynamique et liste chaînée

Objectifs

— Comprendre et savoir utiliser les pointeurs
— Savoir implanter une structure de données dynamique simplement chaînée

Exercice 1 En s'appuyant sur une schématisation des pointeurs, décrire l'effet du programme ci-après. On signalera les instructions erronées et on les ignorera.

```
1  PROCÉDURE Exercice_Pointeurs EST
2
3      --// Définition de types
4      TYPE T_Ptr_Réel EST POINTEUR SUR Réel
5
6      TYPE  T_Complexe EST ENREGISTREMENT
7          Pr : Réel
8          Pi : Réel
9      FIN ENREGISTREMENT
10
11     TYPE T_Ptr_Complexe EST POINTEUR SUR T_Complexe
12
13     --// Déclaration des variables
14     p1 : T_Ptr_Réel
15     p2 : T_Ptr_Réel
16     q1 : T_Ptr_Complexe
17     q2 : T_Ptr_Complexe
18
19 DÉBUT
20     p1 <-- NULL           -- 1
21     p2 <-- NEW Réel       -- 2
22
23     p1^ <-- 1.1           -- 3
24     p2^ <-- 2.2           -- 4
25
26     p1 <-- p2             -- 5
27
28     ÉCRIRE (p1^)          -- 6
29
30     p2^ <-- 3.3           -- 7
31     ÉCRIRE (p1^)          -- 8
32
33     q1 <-- NEW T_Complexe -- 9
34     q1^.Pr <--p1^         -- 10
35     q1^.Pi <-- p2^        -- 11
36     ÉCRIRE (q1^)          -- 12
37     q2 <-- q1             -- 13
38     ÉCRIRE (q2^.Pi)       -- 14
39
40 FIN Exercice_Pointeurs
```

Exercice 2 : Liste simplement chaînée

Considérons le type `T_Liste` définissant une liste chaînée d'entiers.

```
1  TYPE T_Liste EST POINTEUR SUR T_Cellule
2
3  TYPE T_Cellule EST ENREGISTREMENT
4      Élément : Entier          -- Élément rangé dans la cellule
5      Suivante : T_Liste_Entier  -- Accès à la cellule suivante
6  FIN ENREGISTREMENT
```

1. Spécifier, tester et implanter les sous-programmes suivants. Par tester, on entend identifier les cas de test à faire et les données de test correspondantes. On adoptera un style programmation défensive.

1. Initialiser une liste. Cette liste est alors vide.
2. Ajouter un entier en début d'une liste.
3. Obtenir l'entier en début d'une liste.
4. Obtenir la taille (le nombre d'entiers) d'une liste (version itérative et version récursive).
5. Afficher les entiers d'une liste (version itérative et version récursive). On affichera la liste sous la forme suivante (exemple avec la liste `[1, 3, 1, 2]`) :

   ```
   -->[1]-->[3]-->[1]-->[2]--E
   ```

6. Indiquer si un entier e est présent dans une liste (version itérative et version récursive).
7. Supprimer un entier e dans une liste (version itérative et version récursive).
8. Obtenir l'adresse de (le pointeur sur) la première cellule d'une liste qui contient l'entier e.
9. Insérer un entier e après la première occurrence d'un entier f dans une liste.
10. Obtenir l'entier en position i dans une liste. Le premier entier est à la position 0.
11. Supprimer l'entier à la position i dans une liste.

Exercice 3 : Module Liste

Les sous-programmes précédents peuvent être regroupés au sein d'un module générique.
1. Définir l'interface d'un module générique `P_Liste` permettant de gérer des listes chaînées linéaires simples avec encapsulation.
2. Donner la structure générale du corps de ce module.