Méthodologie de la Programmation

1ère année apprentissage (session 1)
Examen
28 Novembre 2014.
Sans documents. Durée 2h.

Remarques :

Tous les programmes seront donnés en langage algorithmique. Ils devront respecter scrupuleusement **TOUTES les normes de bonne programmation**.

Les 2 exercices sont indépendants ;

1 Exercice : La suite de Robinson

On désire construire les éléments d'une suite appelée suite de Robinson.

Le premier terme de la suite de Robinson est posé comme égal à 0. L'élément u_{n+1} de la suite de Robinson se construit en comptant combien de fois chaque chiffre, dans l'ordre de 9 à 0, apparaît dans l'élément u_n. Si un chiffre n'apparaît pas, il n'est pas pris en compte.

- Le premier terme de la suite est 0 : u_0 vaut 0
- u_0 comporte juste **un "0"**, par conséquent le terme suivant u_1 vaut 10
- u_1 est composé d'**un "1" et d'un "0"**, par conséquent le terme suivant u_2 vaut 1110
- u_2 est composé de **trois "1" et d'un "0"**, par conséquent le terme suivant u_3 vaut 3110
- u_3 est composé d'**un "3" et de deux "1" et d'un "0"** , par conséquent le terme suivant u_4 vaut 132110
- u_4 est composé d' **un "3", d'un "2", de trois "1", et d'un "0"** , par conséquent le terme suivant u_5 vaut 13123110 ...

Chaque élément de la suite est caractérisé par une valeur que l'on conservera dans un tableau d'entiers et par un nombre réel d'éléments à conserver aussi . La notation (a, b, ..., c) signifie que T(1) vaut a, T(2) vaut b, ...et T(tailleRéelle) vaut c. Ainsi :

- u_0 est un tableau de 1 élément contenant la valeur 0, ce que l'on notera : u_0 = (0) (nombre d'éléments = 1).
- u_1 est un tableau ayant 2 valeurs la 1ère étant 1, la 2ème étant 0 ce que l'on notera : u_1 = (1,0) (nombre d'éléments = 2)
- u_2 est un tableau ayant 4 valeurs : u_2 = (1,1,1,0) (nombre d'éléments = 4)
- u_3 est un tableau ayant 4 valeurs : u_3 = (3,1,1,0) (nombre d'éléments = 4)
- u_4 est un tableau ayant 6 valeurs : u_4 = (1,3,2,1,1,0) (nombre d'éléments = 6)
- u_5 est un tableau ayant 8 valeurs : u_5 = (1,3,1,2,3,1,1,0) (nombre d'éléments = 8)
- ...

Le type de chaque élément u_i de la suite est donc défini par :

```
NMAX : constante ENTIER = 100
type TVALEURS est Tableau(1..NMAX) de ENTIER
type ELTSUITE est  enregistrement

    V : TVALEURS   -- tableau des valeurs d'un ui
    N : ENTIER -- le nombre de valeurs

fin enregistrement
```

1. Donner l'en-tête d'un sous-programme "suivant" qui permet de calculer u_{i+1} à partir de u_i.
2. Raffiner en langage algorithmique un programme **ITERATIF** affichant les N premiers éléments de la suite de Robinson.
3. Raffiner le problème en langage algorithmique, depuis le niveau R0 jusqu'au niveau Rn correspondant à l'algorithme final, écrit en **ITERATIF**.

2 Exercice : Liste circulaire et application au problème de Josephe

Définition d'une liste circulaire

Une liste circulaire à droite non vide est une liste qui est connue par une référence sur un de ses éléments. Cette référence est appelée position courante (Fig. 1).

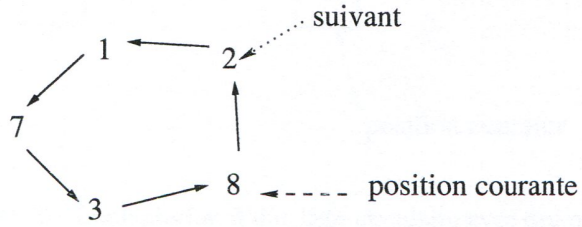
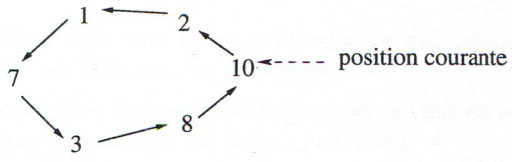
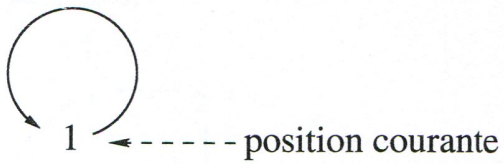


FIGURE 1 – Position courante dans la liste non vide L1

L'insertion d'un élément dans une liste non vide se fait à droite de la position courante. La position courante devient la référence sur l'élément inséré (Fig. 2 (a)).



(a) Résultat de l'insertion de 10 dans L1



(b) Résultat de l'insertion de 1 dans liste vide

FIGURE 2 – Insertion

Si la liste est vide, l'insertion crée une liste dont la position courante est sur son élément unique (Fig. 2 (b)).

2.1 Problème 1

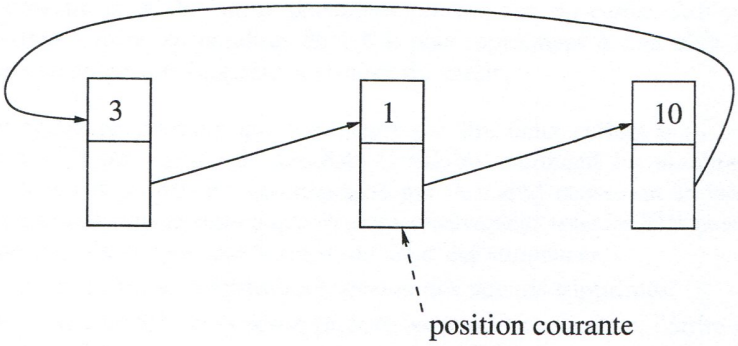


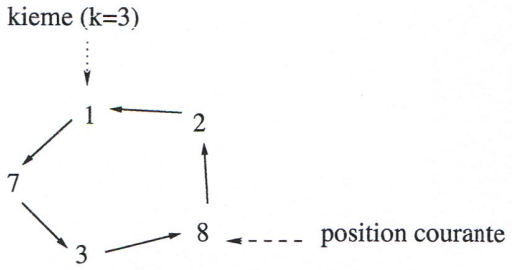
FIGURE 3 – Implantation d'une liste circulaire avec des pointeurs

Une liste circulaire à droite peut être implantée comme une liste simple (Fig. 3).

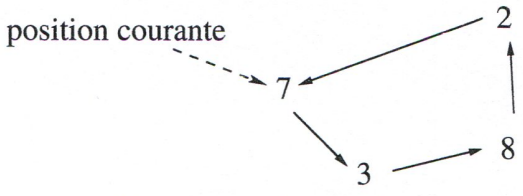
La liste circulaire vide est représentée par le pointeur *null*.

Vous trouverez à la fin du sujet une **partie** de la spécification d'un paquetage qui permet de manipuler ce type de liste circulaire (Fig. 5).

1. Donner la définition type `liste_circulaire` qui permettra d'implanter la liste circulaire en utilisant des pointeurs,
2. Raffiner et implanter la procédure `supprimer_kieme` en respectant les spécifications du paquetage ; un exemple est donné par la Fig. 4



(a) Situation avant suppression



(b) Situation après suppression

FIGURE 4 – Appel de supprimer_kieme avec $k = 3$

2.2 Problème 2 : Application avec le problème de Josèphe

On considère un ensemble de n personnes placées sur un cercle. Les personnes énoncent dans l'ordre les nombres de 1 à k puis reprennent à 1 et ainsi de suite. Toutes les personnes qui énoncent k sortent du cercle.

1. Ecrire un programme qui commence par lire deux entiers n ($n > 0$) et k ($k >= 1$) puis construit une liste circulaire contenant les nombres de 1 à n (dans cet ordre). En commençant par le noeud contenant la valeur 1, le programme doit ensuite supprimer successivement tous les $k^{ème}$ noeuds de la liste jusqu'à ce que tous les noeuds aient été supprimés.

   Il affiche au fur et à mesure les valeurs des noeuds supprimés.

   Exemple : n=8 et k=3 provoqueront les suppressions dans l'ordre suivant : 3,6,1,5,2,8,4,7

   On utilisera les fonctionnalités offertes par le paquetage `pack_liste_circulaire`.

paquetage

```
package pack_liste_circulaire is

  -- type liste_circulaire
  type liste_circulaire is private;

  -- fonction est_vide :
  -- indique si la liste est vide
  -- parametres :
  --   l : liste_circulaire (D)
  -- retour : booleen
  -- pre : aucune
  -- post : aucune
  ------------------------------------------------------------
  function est_vide(l : in liste_circulaire) return boolean;

  ...

  -- fonction creer_liste_1_n :
  -- cree une liste contenant les entiers de 1 a n
  -- parametres :
  --   n : in integer (D)
  -- retour : liste_circulaire
  -- pre : n > 0
  -- post : not est_vide(creer_liste_1_n(m))
  ------------------------------------------------------------
  function creer_liste_1_n(n : in integer) return liste_circulaire;

  -- procedure supprimer kieme:
  -- supprime la kieme valeur a partir de la position courante et met cette
  -- valeur en resultat dans v
  --   si la liste resultat n'est pas vide, la nouvelle position courante est
  --   sur l'element qui etait a droite de la valeur supprimee
  --   sinon la position courante est nulle
  -- parametres :
  --   p_courante : liste_circulaire (D/R)
  --   k : integer (D)
  --   valeur : out integer (R)
  -- pre : la liste n'est pas vide
  --       k >= 1
  -- post : aucune
  ------------------------------------------------------------
  procedure supprimer_kieme(p_courante : in out liste_circulaire;
                            k : in integer; v : out integer);

  private

  ....

end pack_liste_circulaire;
```

FIGURE 5 – Spécification du paquetage (extraits)