

A12 : Représentation chaînée des séquences (corrigés)

EXERCICE 1 : « Lecture » d'un algorithme itératif, avec représentation chaînée, et « écriture » d'un algorithme itératif.

Arranger : l'action (la donnée-résultat T : une AdDoublet)

Lexique : F, Q, P, C, S : des AdDoublets

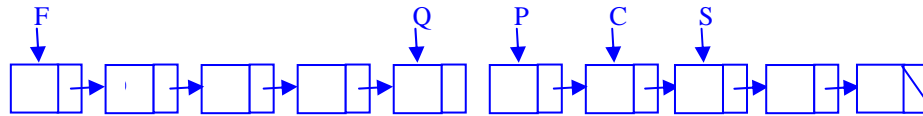
Algorithme :

Allouer(F) ; Q ← F

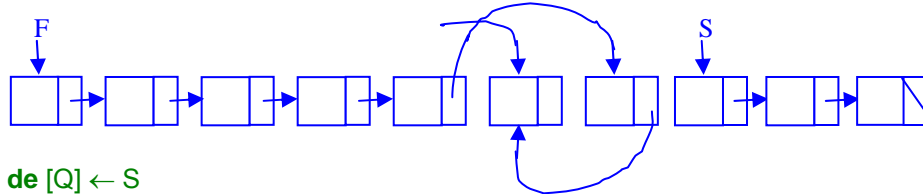
S ← T

tantque (S ≠ Nil) **et puis** (suc de [S] ≠ Nil) :

P ← S ; C ← suc de [P] ; S ← suc de [C]



suc de [Q] ← C ; suc de [C] ← P ; Q ← P



suc de [Q] ← S

T ← suc de [F] ; libérer (F)

Question 1.1 (assertion significative à l'entrée du **tantque**) : S désigne un élément de rang impair, premier élément d'un couple existant.

Question 1.2 (schéma du **tantque**) : Parcours, en modèle 2, des couples d'éléments.

Amorcer : S ← T

Avancer : P ← S ; C ← suc de [P] ; S ← suc de [C]

EstDern : (S = Nil) **oualors** (suc de [S] = Nil)

CpleCour : < P, C >

Question 1.3 (invariant du **tantque**) : La liste chaînée <F, Q>, avec un Fictif F en tête et un pointeur Q sur le dernier élément, représente l'arrangement des éléments initiaux de T à Q (en nombre pair) où tout élément de rang pair $2p$ a été placé devant l'élément de rang impair $2p-1$.

Question 1.4 (action Arranger') :

Arranger' : l'action (les données-résultats T : un tableau sur $[1...N]$ d'entiers ;
L : un entier entre 0 et N)

Lexique :

i, P, C : des entiers entre 1 et N+1

Algorithme :

i ← 1

{ Amorcer }

tantque i < L : { Tous les couples de T[1...i-1] ont été permutés }

{ i désigne un élément de rang impair, premier élément d'un couple existant }

P ← i ; C ← P+1 ; i ← C+1 { Avancer }

T[P] ↔ T[C] { Traiter }

Autre écriture possible :

i ← 1

tantque i < L : T[i] ↔ T[i+1] ; i ← i+2

EXERCICE 2 : Concaténation de deux listes chaînées.

Question 2.1 : **selon** T₁
 T₁ = nil : T ← T₂
 T₁ ≠ nil : T ← T₁
 pc ← T₁ ; **tantque** suc de [pc] ≠ nil : pc ← suc de [pc]
 suc de [pc] ← T₂

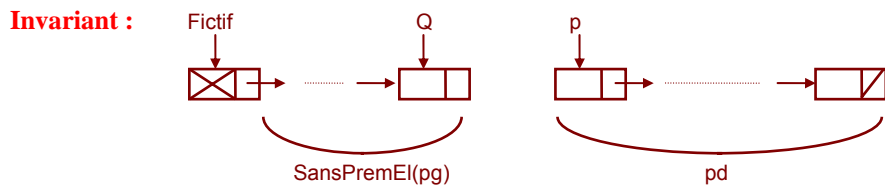
Question 2.2 : T ← T₁
 pc ← T₁ ; **tantque** suc de [pc] ≠ nil : pc ← suc de [pc]
 suc de [pc] ← suc de [T₂]
 libérer(T₂)

Question 2.3 : **selon** T₁
 T₁ = nil : T ← T₂ ; S ← S₂
 T₁ ≠ nil : T ← T₁ ; S ← S₂
 pc ← T₁ ; **tantque** suc de [pc] ≠ S₁ : pc ← suc de [pc]
 suc de [pc] ← T₂
 libérer(S₁)

Exercice 3 :

Supprimer_PremEl : **l'action (la donnée-résultat** Tête : une AdDoublet)
 { état initial : Tête désigne une séquence d'éléments S représentée par chaînage ;
 état final : Tête désigne la séquence d'éléments SansPremEl(S) représentée par chaînage }

si Tête ≠ nil :
 Fictif ← Tête ; Q ← Fictif ; x ← info de [Tête] {le premier élément sert de fictif}
 p ← suc de [Tête]
 tantque p ≠ nil : {mémorisation de tous les éléments ≠ x}
 SucP ← suc de [p]
 selon info de [p], x
 info de [p] ≠ x : suc de [Q] ← p ; Q ← p {insertion en queue}
 info de [p] = x : libérer (p)
 p ← SucP
 suc de [Q] ← nil ; Tête ← suc de [Fictif] ; libérer (Fictif)



Commentaires :

- Le traitement n'est fait que si la séquence n'est pas vide.
- Les deux listes de l'algorithme doivent être mises en évidence :
 - . la liste résultat comporte un fictif et un pointeur de queue de liste, car on y insère "en queue".
 - . la liste donnée est "classique" ; son traitement est une suppression en tête.

Exercice 4 :

Lister_Sommes : l'action (la donnée-résultat Tête : une AdDoublet)

{ état initial : T désigne une séquence d'entiers S représentée par chaînage ;

état final : T désigne la séquence d'entiers SommesSousSeq(S) représentée par chaînage }

allouer (Fictif) ; Q ← Fictif

p ← Tête

tantque p ≠ nil : {p désigne le premier élément d'une sous-séquence croissante}

Σ ← info de [p]

itérer

prec ← p ; p ← suc de [p]

arrêt : p = nil **oualors** info de [p] < info de [prec]

libérer (prec)

Σ ← Σ + info de [p]

{parcours partiel :

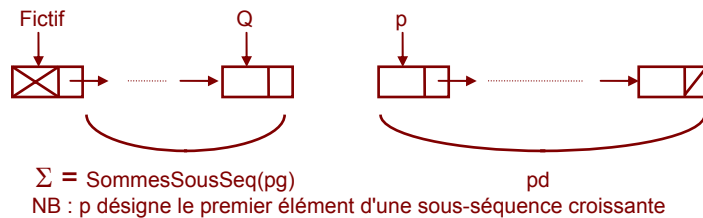
calcul dans S de la somme des éléments de la sous-séquence croissante}

info de [prec] ← Σ ; suc de [Q] ← prec ; Q ← prec

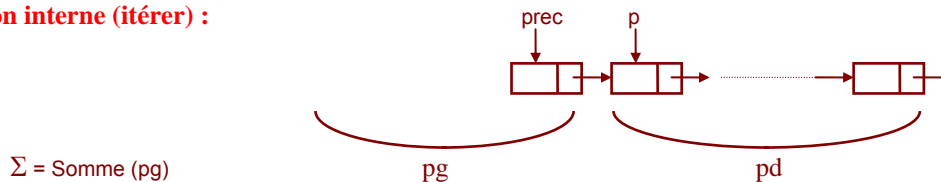
suc de [Q] ← nil ; T ← suc de [Fictif] ; libérer (Fictif)

Invariants :

- itération externe (tantque) :



- itération interne (itérer) :



Commentaires :

- Les deux itérations mettent en évidence la décomposition en sous-séquences croissantes (niveau d'abstraction).
- On conserve un élément de pg (désigné par prec) pour y ranger la somme calculée.