Département STPI Jeudi 4 avril 2013

I3 - Algorithmique

Durée : *1h30*
Documents autorisés : **AUCUN** (calculatrice comprise) [1]

Remarques :
- Veuillez lire attentivement les questions avant de répondre.
- Le barème donné est un barème indicatif qui pourra évoluer lors de la correction.
- Rendez une copie propre.
- N'utilisez pas de crayon à papier.

L'objectif de cet examen est de proposer des algorithmes de traitement du signal. Vous n'appliquerez pas le principe d'encapsulation.

Un signal est une suite de valeurs numériques récupérées à une certaine fréquence. On décide de représenter un signal à l'aide du type suivant :

Type Signal **= Structure**
valeurs : **Tableau[**1..MAX**] de Reel**
nbValeurs : **Naturel**
frequence : **NaturelNonNul**
finstructure

1 Question de cours (3 points)

1. La constante `MAX` est-elle explicite ou implicite ? À quoi sert elle dans le type `Signal` ?
2. Quel est le rôle du champ `nbValeurs` ?
3. Proposez une traduction Pascal de ce type.

Solution proposée :
Voir cours

2 Valeur minimale et maximale d'un signal (4,5 points)

On veut proposer une procédure permettant de calculer la valeur minimale et maximale d'un signal.
- **procédure** calculerValeursMinEtMax **(E** s : Signal,**S** valMin, valMax : **Reel)**

1. Faut il ajouter une précondition à cette signature ? Si oui laquelle. Justifiez.
2. Pourquoi les paramètres formels *valMin* et *valMax* ont un passage de paramètre en sortie ?
3. Proposez l'algorithme de cette procédure.

Solution proposée :
procédure calculerValeursMinEtMax **(E** s : Signal,**S** valMin, valMax : **Reel)**
⌊**précondition(s)** s.nbValeurs > 0
Déclaration i : **NaturelNonNul**
debut
valMin $\leftarrow$ s.valeurs[1]

1. Sauf les dictionnaires pour les étudiants non francophones

```
    valMax ← s.valeurs[1]
    pour i ←2 à s.nbValeurs faire
        si s.valeurs[i]<valMin alors
            valMin ← s.valeurs[i]
        sinon
            si s.valeurs[i]>valMax alors
                valMax ← s.valeurs[i]
            finsi
        finsi
    finpour
fin
```

3 Valeur supérieure ? (3 points)

Proposez une fonction permettant de savoir si un signal possède au moins une valeur strictement supérieure à une valeur *v* donnée.

Solution proposée :

```
fonction valeurSuperieure (s : Signal, v : Reel) : Booleen
    Déclaration  i : Naturel
                 resultat : Booleen
debut
    resultat ← FAUX
    i ← 1
    tant que non resultat et i≤s.nbValeurs faire
        si s.valeur[i]>v alors
            resultat ← VRAI
        sinon
            i ← i+1
        finsi
    fintantque
    retourner resultat
fin
```

4 Lissage d'un signal (9,5 points)

L'objectif ici est de proposer une fonction (`signalLisse`) qui lisse un signal. Une telle fonction calcule la *ième* valeur du nouveau signal comme étant la moyenne des valeurs du signal initial à l'intérieur d'une fenêtre (de taille impaire donnée, supérieure ou égale à 1) centrée en *i* (cf. figure 1). Attention pour les premières et dernières valeurs, seules les valeurs dans la fenêtre sont prises en compte.

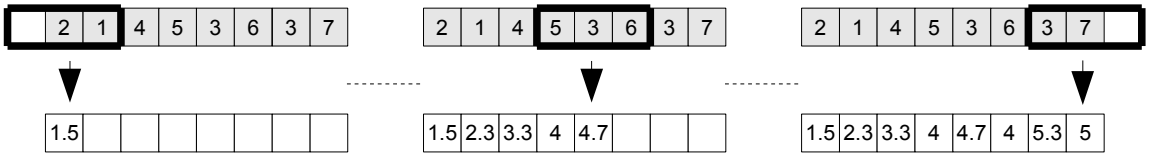


FIGURE 1 – Lissage d'un signal avec une fenêtre de taille 3

4.1 Analyse descendante (4 points)

La figure 2 propose une analyse descendante de ce problème sachant que chaque boite en pointillé représente une entrée ou une sortie d'une opération.

Recopiez cette analyse descendante sur votre copie en remplissant les rectangles en pointillé.

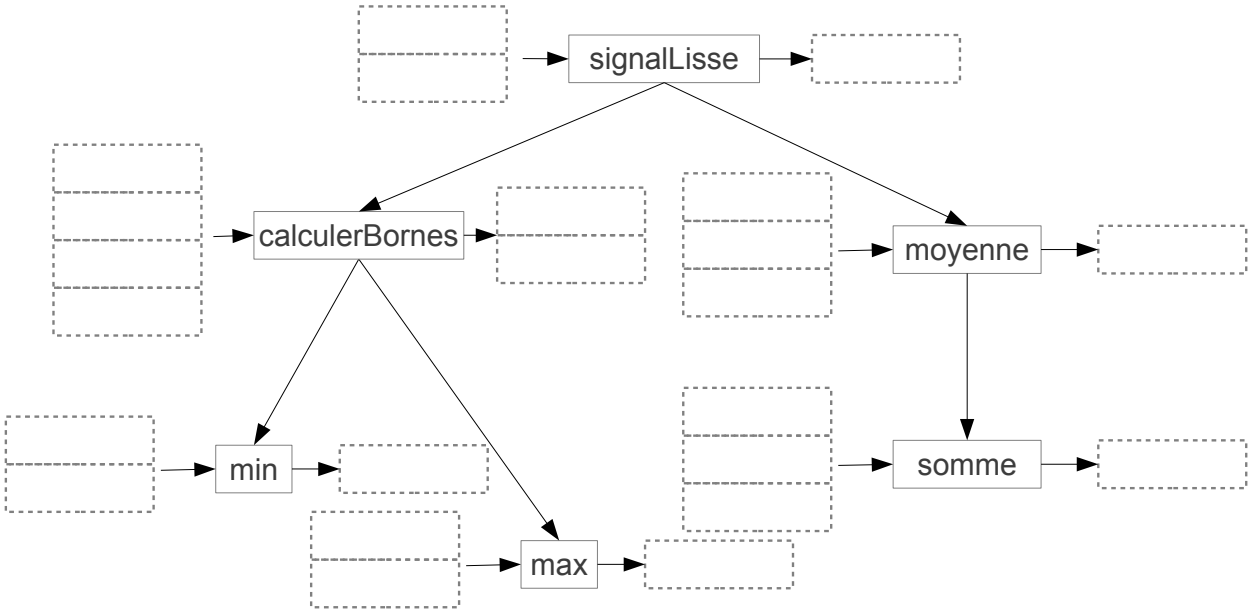
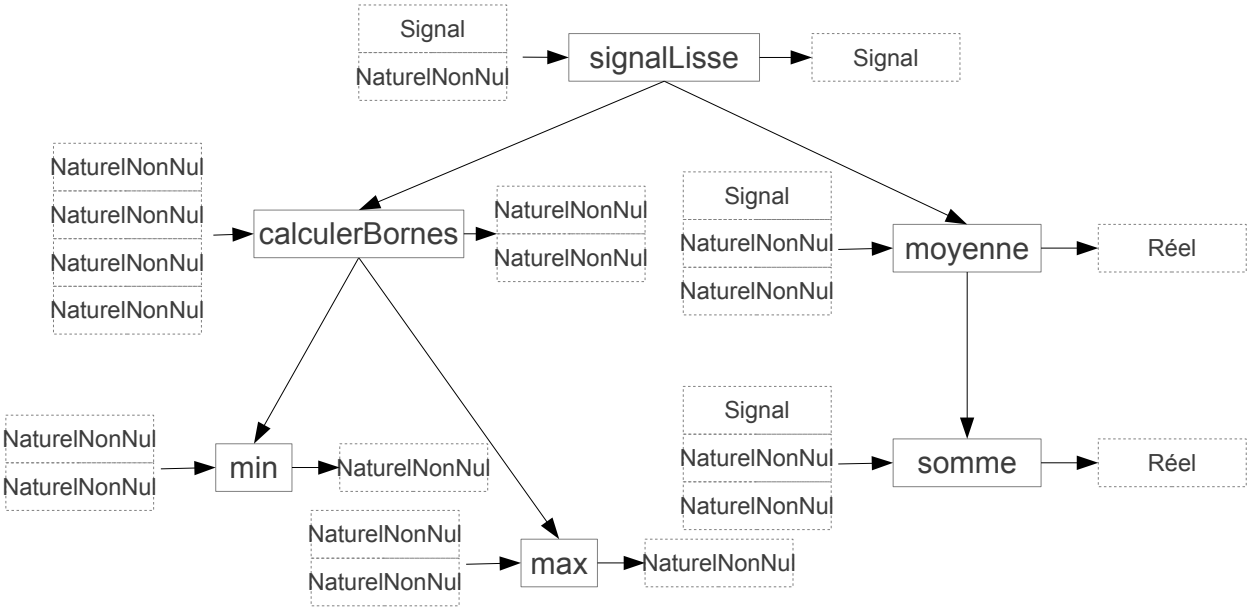


FIGURE 2 – Analyse descendante permettant de lisser un signal

Solution proposée :



4.2 Conception préliminaire (2 points)

Donnez la signature des fonctions ou procédures correspondant aux opérations de l'analyse descendante.

Solution proposée :

- **fonction** signalLisse (s : Signal, tFenetre : **NaturelNonNul) :** Signal

  ⌊**précondition(s)** estImpair(tFenetre)

- **procédure** calculerBornes **(E** i, tFenetre, borneMin, borneMax, **S** borneInf, borneSup : **NaturelNonNul)**

  ⌊**précondition(s)** estImpair(tFenetre)

- **fonction** min (a,b : **NaturelNonNul) : NaturelNonNul**
- **fonction** max (a,b : **NaturelNonNul) : NaturelNonNul**
- **fonction** moyenne (s : Signal, borneInf, borneSup : **NaturelNonNul) : Reel**
- **fonction** somme (s : Signal, borneInf, borneSup : **NaturelNonNul) : Reel**

4.3 Conception détaillée (3,5 points)

Donnez l'algorithme de la fonction ou de la procédure correspondant à l'opération `signalLisse`.

Solution proposée :

fonction signalLisse (unSignal : Signal, tFenetre : **NaturelNonNul) :** Signal

⌊**précondition(s)** estImpair(tFenetre)

Déclaration resultat : Signal
i, borneInf, borneSup : **NaturelNonNul**

debut
resultat.nbValeurs ← unSignal.nbValeurs
resultat.frequence ← unSignal.frequence
pour i ←1 **à** resultat.nbValeurs **faire**
calculerBornes(i, tFenetre,1 ,resultat.nbValeurs , borneInf, borneSup)
resultat.donnes[i] ← moyenne(unSignal, borneInf, borneSup)
finpour
retourner resultat
fin