Chapitre 3 - Traitement séquentiel des données

Georges-Pierre BONNEAU (cours) - Mica MURPHY (note) - Antoine SAGET (note)

Lundi 12 Novembre 2018

Introduction

Avant - Variables entières : N, i, j - Tableau : T[i]

Maintenant - Modèle abstrait d'accès aux données : on dispose d'un ensemble d'**actions** et de **fonction** permettant d'accéder aux données. - Pas d'accès direct aux données : i -> T[i] pas possible avec le modèle abstrait à venir.

Applications - Données sur des supports ne permettant pas un accès direct (*ex* : support magnétique à bande, données en streaming sur un réseau).

Principe général

1. On initialise la séquence,
2. On accède aux éléments l'un après l'autre dans la séquence,
3. On s'arrêt "proprement" à la fin de la séquence.

Solutions

2 solutions pour repérer les fins de séquences → 2 modèles abtraits d'accès séquentiel : - Modèle 1 : on place une marque (virtuelle) **après** le dernier élément. - Modèle 2 : on place la marque **sur** le dernier élément.

Dernier élément

- Inconvénient du modèle 1 : on est obligé d'accèder à l'élement suivant pour savoir que l'élement précedent était le dernier.
- Avantage du modèle 2 : on sait qu'un élément est le denrier quand on le rencontre.

Cas des séquences vides

- Avantage du modèle 1 : pas de problème
- Inconvénient du modèle 2 : on est obligé de rajouter un élément virtuel, l'amroce, situé avant le 1^{er} élément

Accès aux éléments de la séquence

Fonction ou action	Modèle	Spécification
fonction ElementCourant	1,2	retourne l'élément courant
action Avancer	1,2	permet d'avancer d'une unité dans la séquence
fonction booléenne FinDeSéquence	1	retourne vrai ssi on a dépassé le dernier élément
fonction booléenneEstDernier	2	retourne vrai ssi l'élément courant est le dernier élément de la séquence
action Démarrer	1	nous place sur le 1^{er} élément de la séquence$
action Initialiser	2	nous place sur l'amorce (pas le 1^{er} élément)

Modèle 1	Modèle 2
fonction ElementCourant	fonction ElementCourant
action Avancer	action Avancer
fonction booléenne FinDeSéquence	fonction booléenne EstDernier
action Démarrer	action Initialiser

Création d'une séquence vide

Modèle 1

```
Démarrer
Afficher(FinDeSéquence) # vrai
```

Modèle 2

```
Initialiser
Afficher(EstDernier)    # vrai
```

Accès au 1^{er} élement d'une séquence non vide

Modèle 1

```
Démarrer
Afficher(ElementCourant)
```

Modèle 2

```
Initialiser
Avancer
Afficher(ElementCourant)
```

Parcours d'une séquence non vide

Séquence : `1, 5, 3, 2, 4`

Modèle 1

```
Démarrer  -> EC = 1
Avancer   -> EC = 5
Avancer   -> EC = 3
Avancer   -> EC = 2
Avancer   -> EC = 4
Avancer   -> FinDeSéquence vaut vrai => on ne doit pas accéder à EC on ne doit pas avancer
```

Algorithme

```
Démarrer
Tant que (Non FinDeSéquence) faire
  Afficher(ElementCourant)
  Avancer
```

Modèle 2

```
Initialiser
Avancer     -> EC = 1
Avancer     -> EC = 5
Avancer     -> EC = 3
Avancer     -> EC = 2
Avancer     -> EC = 4
            -> EstDernier vaut vrai => on ne doit pas avancer
```

Algorithme

```
Initialiser
Tant que (Non EstDernier) faire
  Avancer
  Afficher(ElementCourant)
```

Remarque : dans les deux cas on a fait `Avancer` 5 fois

Interdictions

- En modèles 1 et 2 : appeler `Avancer` sans avoir vérifié la valeur de `FinDeSéquence` ou `EstDernier`
- En modèle 1 : appeler `EC` avant d'avoir vérifié `FinDeSéquence`
- En modèle 2 : appeler `EC` avant d'avoir fait `Avancer` au moins une fois

Construction de séquences

On part d'une séquence de caractères, lettres ou espace

Longueur des mots

On va définir la séquence de la longueur des mots

Mot : une séquence contigue de caractères différent de l'espace

Exemple : '_ _ une _ _ _ séquence _' -> {3, 8}

La séquence de caractères est données en **Modèle 1**, pour y accéder on a : - `DemCar` - `AvCar` - `ECCar` - `FdSCar`

Séquence des longueurs des mots en **Modèle 1** et **2** :

Modèle 1

```
___--____------__----_____
   ^        ^     ^    ^ # Modèle 1
  ^     ^       ^       ^  # Modèle 2
```

- **Modèle 1** : j'avance de fin de mot en fin de mot
- **Modèle 2** : j'avance de début de mot en début de mot

```
---_____-----
   ^    ^
   ei   ef
----____----
^
ei
ef
```

```
IgnorerEspaces : une action
{État initial : FdSCar ou alors ECCar est le i-ème caractère}
{État final : FdSCar ou alors ECCar est la 1ère leetre dut mot suivant}

{Algorithme :
  TantQue (Non FdSCar et puis ECCar = espace)
    AvCar
}

{Remarque : rien ne se passe si ECCar n'est pas un espace en état initial}
```

```
CalculerLongueur : une action (le résultat Long : un entier)
{État initial : Non FdSCar et ECCar est la 1ère lettre d'un mot}
{État final :
  - Long vaut la longueur du mot
  ET
  - FdSCar et le mot est le dernier OU ECCar est l'espace qui suit le mot
}

{Algorithme :
  Long <- 1
  Itérer
    AvCar
    Arrêt si FdSCar ou alors ECCar = espace
    Long <- Long + 1
}
```

```
Variables partagées par DemLong, AvLong, ECLong, et FdSLong :
{Lexique :
  - FinLong : un booléen
  - Long : un entier
}
```

```
DemLong : une action
{État initial : indifférent}
{État final :
  - FinLong
  OU
  - Long est la longueur du 1er mot ET (FdsCar OU ECCar est l'espace suivant le 1er mot)
}

{Algorithme :
  DemCar
  IgnorerEspaces
  Selon FdSCar :
    - FdSCar : FinLong <- vrai
    - Non FdSCar :
        CalculerLong(Long)
        FinLong <- faux
```

```
}

AvLong : une action
{État initial :
  - Non FinLong ET Long est la longueur du i-ème mot
  ET
  - FdSCar ou ECCar est l'espaec suivant le i-ème mot
}
{État final :
  - FinLong
  OU
  - Long est la longueur du (i + 1)-ème mot ET (FdsCar OU ECCar est l'espace suivant le (i + 1)-ème mo
}

{Algorithme :
  IgnorerEspaces
  Selon FdSCar :
    - FdSCar : FinLong <- vrai
    - Non FdSCar :
        CalculerLong(Long)
        FinLong <- faux
}

ECLong : la fonction -> un entier
{Algorithme : retourne (Long)}

FdSLong : la fonction -> un booléen
{Algorithme : retourne (FinLong)}
```

Calcul de la somme des longueurs des mots :

```
{Lexique : Somme : un entier}

{Algorithme :
  DemLong
  Somme <- 0

  Tant que Non FdSLong :
    Somme <- Somme + ECLong
    AvLong
}
```

Modèle 2

La séquence des caractère est toujours en modèle 1

On cherche à créer les fonctions `InitLong`, `AvLong`, `ECLong`, `EstDernierLong`

```
Variables partagées par InitLong, AvLong, ECLong, et EstDernierLong :
{Lexique : Long : un entier}

InitLong : une action
{État initial : indifférent}
{État final : FdSCar OU ECCar est la 1ère lettre du 1er mot}

{Algorithme :
  DemCar
  IgnorerEspaces
}
```

```
AvLong : une action
{État initial : Non FdSCar ET ECCar est la 1ère lettre du i-ème mot}
{État final :
  - Long vaut la longueur du i-ème mot
  ET
  - FdSCar OU (ECCar est la 1ère lettre du (i + 1)-ème mot)
}

{Algorithme :
  CalculerLongueur(Long)
  IgnorerEspaces
}

ECLong : une fonction -> entier
{Algorithme : retourner (Long)}

EstDernierLong : une fonction -> un booléen
{Algorithme : retourner (FdSCar)}
```

.