Algorithmique & programmation

Chapitre 2 : Vecteurs

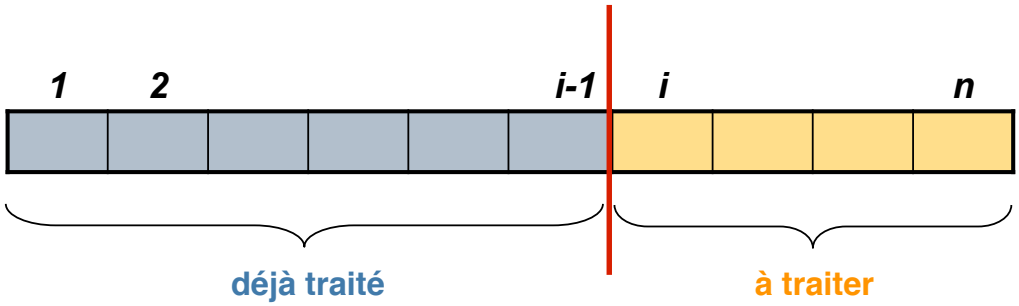
Algorithmes de parcours complet

Somme des éléments d'un vecteur

- *On veut faire la somme de toutes les valeurs contenues dans un vecteur d'entiers*
- Pour faire le raisonnement par récurrence, il faut d'abord trouver **l'hypothèse** !
  - *Faire un dessin*

Somme des éléments d'un vecteur

- Dessin



Somme des éléments d'un vecteur

- On veut faire la somme de toutes les valeurs contenues dans un vecteur d'entiers
- Pour faire le raisonnement par récurrence, il faut d'abord trouver **l'hypothèse** !
  - *Faire un dessin*
  - en français
    - À un instant donné, l'algorithme qui fait la somme des éléments du vecteur à fait la somme des éléments d'un sous-vecteur V[1..i-1]
  - formalisée
    - $s = \sum V[1..i-1]$

Somme des éléments d'un vecteur

- Le raisonnement par récurrence

Hypothèse $s = \sum V[1..i-1]$

Q : Quelles sont les situations intéressantes ?
R : Il y en a deux !

- $i = n+1 \Rightarrow s = \sum V[1..n] \Rightarrow$ résultat = s ✳
- $i \leq n \Rightarrow s := s + V[i] ; i := i + 1 ; \Rightarrow H$

Q : Dans quelle situation retourne-t-on à l'hypothèse ?
R : Lorsque $i \leq n$!

Itération tantque $i \leq n$ faire …

Q : Connaît-on un vecteur pour lequel on peut rendre le résultat sans calcul ?
R : Oui, le vecteur vide !

Initialisation $s := 0 ; i := 1 ; \Rightarrow H$

Somme des éléments d'un vecteur

- Le raisonnement par récurrence

Hypothèse $s = \sum V[1..i-1]$

- $i = n+1 \Rightarrow s = \sum V[1..n] \Rightarrow$ résultat = s ✳
- $i \leq n \Rightarrow s := s + V[i] ; i := i + 1 ; \Rightarrow H$

Itération tantque $i \leq n$ faire …

Initialisation $s := 0 ; i := 1 ; \Rightarrow H$

fonction **somme**

fonction somme (d V[1..n] : vecteur) : entier ;
spécification *{n ≤ 0} ➔ {résultat = Σ V[1..n]}*
    s, i : entier ;
debfonc
    s := 0 ; i := 1 ; *Initialisation*
    tantque i ≤ n **faire** *itération*
        s := s + V[i] ; *cas retournant à l'hypothèse*
        i := i + 1 ;
    finfaire ;
    {i>n, s = Σ V[1..i-1] ⇒ i=n+1, s = Σ V[1..n]}
    retour s ; produire le résultat
finfonc ;

fonction **somme**

- soit V un vecteur d'entier

```
function somme (V : in vecteur) return Integer is
-- {V vecteur vide ou non} -> {résultat = Σ V}
-- il faut retrouver les bornes !!!
  s, i : integer ;
begin
  s := 0 ; i := V'First ;
  while i ≤ V'Last loop
    s := s + V(i) ;
    i := i + 1 ;
  end loop;
  --{i>n, s = Σ V[1..i-1] ⇒ i=n+1, s = Σ V[1..n]}
  retrun s ;
end somme ;
```

1 de V[1..n] {borne inférieure}

n de V[1..n] {borne supérieure}

V(i) remplace V[i]

fonction **somme**

fonction somme (d V[1..n] : vecteur) : entier ;
spécification *{n ≤ 0} ➔ {résultat = Σ V[1..n]}*
s : entier ;
debfonc
    s := 0 ;
    pour i := 1 **haut** n **faire** *itération d'un parcours complet*
        s := s + V[i] ;
    finfaire ;
    retour s ;
finfonc ;

fonction **somme**

- soit V un vecteur d'entier

```
function somme (V : in vecteur) return Integer is
-- {V vecteur vide ou non} -> {résultat = Σ V}
-- il faut retrouver les bornes !!!
  s : integer ;
begin
  s := 0 ;
  for i in V'range loop
    s := s + V(i) ;
  end loop;
  --{i>n, s = Σ V[1..i-1] ⇒ i=n+1, s = Σ V[1..n]}
  retrun s ;
end somme ;
```

i parcours tous les indices