TDA : Chaîne de caractères

Sébastien Jean

IUT de Valence
Département Informatique

v1.0, 1er décembre 2025

IUT Valence
Université Grenoble Alpes

Type de données *Chaîne de caractères*

- Type de données linéaire et dynamique, permettant de représenter une séquence de caractères.
- Possibilité d'**obtenir la taille**, de **savoir si la chaîne est vide**, **d'obtenir un caractère à une position donnée**, d'**ajouter un caractère en fin**

IUT Valence
Université Grenoble Alpes

TDA *Chaîne*

- **Nom** : `Chaîne`
- **Dépendances** : `Booléen`, `Entier`, `Caractère`
- **Opérations** :
  - **Constructeurs** :
    - `chaîne_vide` : $\rightarrow$ `Chaîne`
  - **Transformateurs** :
    - `ajouter_caractère` : `Chaîne` x `Caractère` $\rightarrow$ `Chaîne`
  - **Observateurs** :
    - `taille` : `Chaîne` $\rightarrow$ `Entier`
    - `est_vide` : `Chaîne` $\rightarrow$ `Booléen`
    - `caractère_en` : `Chaîne` x `Entier` $\rightarrow$ `Caractère`
- **Pré-conditions**
  - `caractère_en(ch, i)` $\rightarrow$ `1` $\leq$ `i` $\leq$ `taille(ch)`

IUT Valence
Université Grenoble Alpes

TDA *Chaîne* (suite)

- **Axiomes**
  - `est_vide(chaîne_vide()) = VRAI`
  - `est_vide(ajouter_caractère(ch, c)) = FAUX`
  - `taille(chaîne_vide()) = 0`
  - `taille(ajouter_caractère(ch, c)) = taille(ch) + 1`
  - `caractère_en(ajouter_caractère(ch, c), i) = c,`
    `si i = taille(ch)+1`
  - `caractère_en(ajouter_caractère(ch, c), i)`
    `= caractère_en(ch,i), si i` $\leq$ `taille(ch)`

IUT Valence
Université Grenoble Alpes

Chaîne et pseudo code

- Dans notre *pseudo code*, on suppose que
  - Le constructeur est remplacé par la **déclaration de la variable**
  - L'écriture d'une séquence de caractère entre " " est une expression équivalente une chaîne de caractères contenant tous les caractères
  - Les transformateurs sont des mutateurs
  - Les paramètres sont **passés par copie**

```
VARIABLE ch : Chaîne
VARIABLE i : Caractère

ch ← "Georges Abitbol"
ajouter_caractère(ch,'!')

i ← caractère_en(ch, 3)

 AFFICHER(taille(ch))
```

IUT Valence Université Grenoble Alpes

Exercice

Enoncé du problème

On souhaite connaitre l'indice d'un caractère (s'il est présent dans la chaîne)

Spécification du problème

- **Donnée d'entrée** : `ch`, chaîne (la chaîne où chercher)
- **Donnée d'entrée** : `c`, caractère (le caractère à chercher)
- **Donnée de sortie** : `i`, entier (l'indice trouvé)
- **Pré-condition** : (aucune)
- **Post-condition** :
  - si `i` ≥ 1, alors la première occurence de `c` est en position `i` de `ch`
  - si `i` $= 0$, alors le caractère `c` n'est pas présent dans `ch`

Signature de la fonction

- **indice_de** (ch : chaîne, c : caractère) : entier

Exercice

```
FONCTION indice_de(ch : chaine, c : caractère)
                    : entier

 VARIABLE indice : entier

 POUR indice de 1 A taille(ch) PAR PAS DE 1
  SI caractère_en(ch, indice) = c ALORS
    RETOURNER indice
  FIN SI
 FIN POUR

 RETOURNER 0

FIN FONCTION
```

IUT Valence
Université Grenoble Alpes

Exercice

Enoncé du problème

On souhaite extraire une sous-chaîne d'une chaîne (entre 2 indices inclus)

Spécification du problème

- **Donnée d'entrée** : `ch`, chaîne (la chaîne dans laquelle extraire)
- **Donnée d'entrée** : `s`, entier (l'indice de début)
- **Donnée d'entrée** : `e`, entier (l'indice de fin)
- **Donnée de sortie** : `r`, chaîne (la sous-chaîne extraite)
- **Pré-condition** : $1 \leq s \leq e \leq taille(ch)$
- **Post-condition** : `r` est la chaîne contenant les caractères de `ch` présents entre les indices `s` et `e` inclus

Signature de la fonction

- **sous_chaîne** (ch : chaîne, s : entier, e : entier) : chaîne

Université Grenoble Alpes

Exercice

```
FONCTION sous_chaîne(ch : chaine, s : entier,
                     e : entier) : chaîne

 VARIABLE indice : entier
 VARIABLE r      : chaîne

 POUR indice de s A e PAR PAS DE 1
  ajouter_caractère(r, caractère_en(ch, indice))
 FIN POUR

 RETOURNER r

FIN FONCTION
```

IUT Valence
Université Grenoble Alpes

Exercice

Enoncé du problème

On souhaite produire une chaîne de caractères par concaténation de 2 chaines de caractères.

Spécification du problème

- **Donnée d'entrée** : `ch1`, chaîne (la première chaîne)
- **Donnée d'entrée** : `ch2`, chaîne (la seconde chaîne)
- **Donnée de sortie** : `r`, chaîne (la concaténation des 2)
- **Pré-condition** : (aucune)
- **Post-condition** : `r` est la chaîne contenant les caractères de `ch1` suivis des caractères de `ch2`

Signature de la fonction

- **concaténer** (ch1 : chaîne, ch2 : chaîne) : chaîne

Valence
Université Grenoble Alpes

Exercice

```
FONCTION concaténer(ch1 : chaine, ch2 : chaine) :
    chaîne

 VARIABLE indice : entier
 VARIABLE r      : chaîne

 POUR indice de 1 A taille(ch1) PAR PAS DE 1
  ajouter_caractère(r, caractère_en(ch1, indice))
 FIN POUR

 POUR indice de 1 A taille(ch2) PAR PAS DE 1
  ajouter_caractère(r, caractère_en(ch2, indice))
 FIN POUR

 RETOURNER r

FIN FONCTION
```

IUT Valence Université Grenoble Alpes

Exercice

Enoncé du problème

On souhaite savoir si deux chaînes sont égales.

Spécification du problème

- **Donnée d'entrée** : `ch1`, chaîne (la première chaîne)
- **Donnée d'entrée** : `ch2`, chaîne (la seconde chaîne)
- **Donnée de sortie** : `r`, booléen
- **Pré-condition** : (aucune)
- **Post-condition** : `r` = `VRAI` si `ch1` et `ch2` possèdent les mêmes caractères dans le même ordre, `r` = `FAUX` sinon

Signature de la fonction

- **egales** (ch1 : chaîne, ch2 : chaîne) : booléen

IUT Valence
Université Grenoble Alpes

Exercice

```
FONCTION égales(ch1 : chaine, ch2 : chaine) : booléen

 VARIABLE indice : entier

 SI taille(ch1) ≠ taille(ch2) ALORS
   RETOURNER FAUX
 FIN SI

 POUR indice de 1 A taille(ch1) PAR PAS DE 1
   SI     caractère_en(ch1, indice)
        ≠ caractère_en(ch2, indice) ALORS
     RETOURNER FAUX
   FIN SI
 FIN POUR

 RETOURNER VRAI

FIN FONCTION
```

IUT Valence Université Grenoble Alpes

Exercice

Enoncé du problème

On souhaite savoir si une sous-chaîne est présente dans une chaîne de caractères.

Spécification du problème

- **Donnée d'entrée** : `ch1`, chaîne (la première chaîne)
- **Donnée d'entrée** : `ch2`, chaîne (la seconde chaîne)
- **Donnée de sortie** : `r`, booléen
- **Pré-condition** : `ch2` n'est pas vide
- **Post-condition** : `r` = `VRAI` si `ch2` est présente dans `ch1`, `r` = `FAUX` sinon

Signature de la fonction

- **est_sous_chaîne** (ch1 : chaîne, ch2 : chaîne) : booléen

Valence
Université Grenoble Alpes

Exercice

```
FONCTION est_sous_chaîne(ch1 : chaine, ch2 : chaine)
         : booléen

 VARIABLE i : entier
 VARIABLE r : chaîne

 SI taille(ch1) < taille(ch2) ALORS
   RETOURNER FAUX
 FIN SI

 POUR i de 1 A taille(ch1)-taille(ch2)
 PAR PAS DE 1
  SI egales(sous_chaine(ch1, i, i+taille(ch2)-1, ch2))
       ALORS
    RETOURNER VRAI
  FIN SI
 FIN POUR
 RETOURNER FAUX
FIN FONCTION
```

IUT Valence
Université Grenoble Alpes

Fin !