Algorithmique et Structures de Données 1
CH5: Les chaînes de caractères

Enseignant: Fethi Mguis
Sections: LGLSI1/LIRIS1

A.U: 2023/2024

1 Le type caractère

1.1 Définition

Ce type s'applique à tous les caractères du code ASCII (Americain Standard Code for Information Interchange). La liste comprend :

- Les lettres : 'A'...'Z', 'a'...'z'
- Les chiffres : '0'...'9'
- Les caractères spéciaux : '/' ; '*' ; ' ?' ; '&' etc.
- Les caractères de contrôles : ¡retour chariot¿ ; ¡Echap¿ ; ¡ctrl¿ etc.

Chaque caractère est défini par son numéro d'ordre unique compris entre 0 et 255 et ordonné selon leur code ASCII : '0' < '1' ... '9' < ... < 'A' < 'B' < ... < 'Z' < ... < 'a' < 'b' < ... < 'z'

- Code ASCII de 'A' = 65
- Code ASCII de 'a' = 97
- Code ASCII de '0' = 48

1.2 Fonctions standards sur les caractères

On dispose d'un ensemble de fonctions prédéfinies qui peuvent être utilisées avec les objets de type carcatère. Les principales fonction sont données dans le Tableau 7.1.

Fonction	**Rôle**	**Exemple**	**Résultat**
ASC(c) ou ORD(c)	Retourne le code ASCII du caractère "c"	i←ASC('B')	i contiendra 66
CAR(i) ou CHR(i)	Retourne le caractère dont le code ASCII est égal à "i"	c←CAR(68)	c contiendra 'D'
SUCC(c)	Retourne le successeur du caractère "c"	c←SUCC('a')	c contiendra 'b'
PRED(c)	Retourne le successeur du caractère "c"	c←SUCC('e')	c contiendra 'd'
MAJUS(c)	Retourne la majuscule du caractère "c"	c←MAJUS('f')	c contiendra 'F'

Table 1 – Fonctions standards pour le type caractère.

Exercice 1 Ecrire un algorithme qui lit un caractère au clavier puis affiche son prédécesseur, son successeur et le code ASCII de son équivalent en majuscule.

```
Algorithme CARACTERE
Var
   C : caractère
Début
  Écrire("Donner un caractère :")
  Lire(C)
  Écrire("Le prédécesseur =", PRED(C))
  Écrire("Le successeur=", SUCC(C))
  Écrire("le code ASCII de son équivalent en majuscule=", ASC(MAJUS(C)))
Fin
```

Exercice 2 Ecrire un algorithme qui lit une lettre au clavier puis affiche s'il s'agit d'une consonne ou d'une voyelle.

```
Algorithme CON_VOY
Var
   C : caractère
Début
  Répéter
  | Écrire("Donner un caractère :")
  | Lire(C)
  Jusqu'à ((C>='A' et C<='Z') OU (C>='a' et C<='z'))
  Si ((MAJUS(C)='A') OU (MAJUS(C)='E') OU (MAJUS(C)='I') OU (MA-
  JUS(C)='O') OU (MAJUS(C)='U') OU (MAJUS(C)='Y')) Alors
  | Écrire ( c, "est une voyelle ")
  Sinon
  | Écrire ( c, "est une consonne ")
  Fin Si
Fin
```

2 Le type chaîne de caractères

2.1 Définition

Une chaîne de caractères est une succession de n caractères avec n compris entre 0 et 255. Si n = 0 on dit que la chaîne est vide.

2.2 Déclaration d'une chaîne

```
Nom_chaîne : chaîne [Lmax]
```

Exemple

```
Var
ch : chaîne
chn : chaîne[10]
```

La variable "ch" peut contenir jusqu'au 255 caractères alors que "chn" peut contenir au maximum 10 caractères.

```
ch←"" vide
ch←" " un espace
ch←"Alexandrie "
chn←ch
ch←"Algorithmique et structure de données 2 "
```

2.3 Opérations sur les chaînes de caractères

2.3.1 La concaténation

C'est l'assemblage de deux chaînes de caractères en utilisant l'opérateur "+"

Exemple

```
CH1←"Algorithmique "
CH2←"et structure de données 2"
CH3←CH1 + CH2
```

CH3 contiendra "Algorithmique et structure de données 2 "

2.3.2 Les opérateurs relationnels

Il est possible d'effectuer une comparaison entre deux chaînes de caractères, le résultat est de type booléen. La comparaison se fait caractère de la gauche vers la droite selon le code ASCII.

Exemple

- 'w' > 'W' vrai car code ASCII de 'w' = 119 et code ASCII de 'W' = 87
- "programme" > "programmation" faux car 'e' > 'a'
- " = ' ' est fausse car le vide est différent du caractère espace code ASCII ' ' = 32

2.3.3 Accès à un caractère dans une chaîne

On pourra accéder en lecture ou en écriture au ième caractère d'une chaîne CH en utilisant la notation CH[i].

Exemple

```
Var  CH : chaîne[10]
CH←"Structure"
CH[1] donne 'S'
CH[6] donne 't'
CH[10]←'s' CH devient "Structures"
```

2.4 Les fonctions et les procédures standards sur les chaînes

2.4.1 Les fonctions standards (Les plus utilisées)

Concaténation Concat(ch1, ch2, ..., chN)
Fonction qui retourne la concaténation de ch1, ch2, ..., chN.
C'est l'équivalent de ch1 + ch2 + ... + chN

Exemple

```
jj←"16"
mm←"01"
aa←"2006"
DATE←Concat(jj, "/", mm, "/", aa) donne 16/01/2006
```

Longueur d'une chaîne Long (ch)
Fonction qui retourne un entier représentant la longueur en caractères de la chaîne "ch".

Exemple

```
L←Long ("Bonjour") donne 7
L←Long (" Bonjour") donne 8
```

Copie Syntaxe :
Copie (ch, P, N) ou **souchaine(ch , P , N)**
Fonction qui retourne une sous chaîne d'une longueur "N" à partir de la position "p" dans "ch".
Exemple :

```
CH1←"Algorithmique et structure de données 2 "
CH2←Copie (ch1, 1, 4)
```

CH2 contiendra "Algo"

Position POS (ch1, ch2)
Fonction qui retourne la position de la première occurrence de la chaîne ch1 dans la chaîne ch2 sinon elle retourne 0.

Exemple :

```
CH1←"Technicien"
CH2←"cien"
i←POS(ch2, ch1) donne 7
j←POS("tec ", ch1) donne 0
```

Comparaison Compare (ch1, ch2)
Fonction qui retourne :

- Zéro : Si CH1 et CH2 sont égaux.
- Un nombre négatif : Si CH1 précède CH2 dans l'ordre alphabétique.
- Un nombre positif : Si CH2 précède CH1 dans l'ordre alphabétique.

Exemple

```
ch1←"aa"
ch2←"AB"
R←Compare(ch1, ch2)
```

R contient un nombre positif

2.4.2 Les procédures standards (Les plus utilisées)

effacement Efface (ch, p, n)
Procédure qui enlève "n" caractères de "ch" à partir de la position "p".

Exemple

```
mot←"CD ROM"
Efface(mot, 3, 4)
```

mot devient "CD"

Insertion Insère (ch1, ch2, p)
Procédure qui insère la chaîne "ch1" dans la chaîne "ch2" à partir de la position "p". Le caractère numéro "p" et les suivants sont décalés vers la droite.
Exemple :

```
ch1←"DA"
ch2←"DIC"
Insère (ch1, ch2, 3)
```

ch2 devient "DIDAC"

Conversion
Nombre $-->$ Chaine
Convch (n, ch1)
Procédure qui convertit un nombre "n" en une chaîne de caractère dans la variable "ch1".
Exemple :
Convch (2006, ch1) ch1 devient "2006"
Chaine $-->$ Nombre
Valeur (ch, n, erreur)
Procédure qui convertit une chaîne "ch" en une valeur numérique dans la variable numérique "n" "erreur" est une variable de type entier qui contiendra 0 si la conversion s'est déroulée sans erreur, sinon elle contiendra le numéro (la position) du caractère qui a déclenché l'erreur.
Exemple :

```
ch←"2006"
Valeur ( ch, n, erreur )
```

n contient le nombre 2006
erreur contient 0 (aucune erreur)

```
ch←"15/01/2006"
Valeur ( ch, n, erreur )
```

n contient le nombre 15
erreur contient 3 (erreur à la position 3 dû au caractère "/")

Exercice Ecrire un algorithme qui lit une chaîne de caractères et vérifie si cette chaîne est un palindrome ou non. Un palindrome est un mot qui peut être lu indifféremment de droite à gauche ou de gauche à droite (Exemples : "AZIZA", "LAVAL", ...)

```
Algorithme PALINDROME
Var
   CH : chaîne
i, L : entier
Pal : booléen
Début
  Écrire ("Donner une chaîne :")
  Lire (CH)
  L←Long(CH)
  i←1
  Pal←vrai
  Tant que (i <= L/2 et Pal = vrai) faire
    Si (CH[i] = CH[L - i + 1]) Alors
      i←i + 1
    Sinon
      Pal←faux
    Fin Si
  Fin Tq
  Si (Pal = vrai) Alors
    Écrire (CH, " est un palindrome ")
  Sinon
    Écrire (CH, " n'est pas un palindrome ")
  Fin Si
Fin
```