Caractères et chaînes de caractères

1 Les caractères

1.1 Définition

Les *caractères* sont un ensemble de symboles comprenant les 26 lettres de l'alphabet latin en versions minuscules et majuscules, les 10 chiffres de 0 à 9, les différents symboles de ponctuation, l'espace, et bien d'autres symboles encore. Ils sont au nombre de 256 au total.

En Pascal, ils forment le type de données `CHAR`.

Il est possible de déclarer des constantes de type `CHAR`. On peut par exemple définir la constante

```
const
   ESPACE = '␣';
```

Les constantes litérales de type `CHAR` sont désignées entre deux apostrophes.

On peur aussi déclarer des variables de type `CHAR`.

```
var
   c : CHAR;
```

Il est possible de lire une donnée de ce type et de l'écrire au moyen des procédures **`readln`** et **`write(ln)`**. Le programme ci-dessous lit un caractère saisi au clavier, l'affecte à la variable `c`, puis l'écrit à l'écran.

```
begin
   readln(c);
   writeln(c);
end.
```

1.2 Fonctions **ord** et char

Les caractères sont numérotés de 0 à 255. Par exemple, le caractère `ESPACE` porte le numéro 32, le caractère `A` porte le numéro 64 et le caractère `a` porte le numéro 97.

En Pascal, la fonction prédéfinie **`ord`** associe au caractère qu'on lui passe en paramètre le numéro associé à ce caractère. Par exemple, le programme

```
begin
   writeln(ord('␣'));
   writeln(ord('A'));
   writeln(ord('a'));
end.
```

affiche à l'écran les trois nombres 32, 65 et 97.

Inversement, la fonction **`char`** associe à l'entier (compris entre 0 et 255) qu'on lui passe en paramètre le caractère ayant pour numéro cet entier. Le programme

```
begin
   writeln(char(32));
   writeln(char(65));
   writeln(char(97));
end.
```

affiche les trois caractères `ESPACE`, `A` et `a`.

1.3 Comparaison de caractères

Les opérateurs `=`, `<`, `<=`, `>` et `>=` permettent de comparer deux caractères.

Si `c1` et `c2` sont deux expressions de type `CHAR`, alors

1. `c1 = c2` est vrai si et seulement si les deux caractères `c1` et `c2` sont égaux ;

2. `c1 < c2` est vrai si et seulement si le numéro du caractère `c1` est strictement plus petit que celui de `c2` ;
3. `c1 <= c2` est vrai si et seulement si le numéro du caractère `c1` est inférieur ou égal à celui de `c2` ;
4. etc . . .

2 Les chaînes de caractères

2.1 Définition

Les *chaînes de caractères* sont des séquences finies de caractères. Le nombre de caractères composant une chaîne de caractères est la *longueur* de cette chaîne.

Une chaîne de longueur nulle ne comprend aucun caractère : c'est la *chaîne vide*.

En Pascal, les chaînes de caractères forment le type **STRING**.

Il est possible de déclarer des constantes de type **STRING**. On peut par exemple définir les constantes

```
const
   CHAINE1 = 'TIMO';
   CHAINE2 = 'LEON';
   VIDE    = ''; // chaîne vide
```

Les constantes litérales de type **STRING** sont désignées entre deux apostrophes.

On peur aussi déclarer des variables de type **STRING**.

```
var
   s : STRING;
```

Il est possible de lire une donnée de ce type et de l'écrire au moyen des procédures **readln** et **write(ln)**. Le programme ci-dessous commence par se présenter, puis demande le nom de l'utilisateur, affiche une formule de politesse personnalisée, et finalement informe l'utilisateur du nombre de lettres de son nom.

```
begin
   writeln('Je␣m''appelle␣', CHAINE1, CHAINE2,'.');
   writeln('Et␣vous,␣comment␣vous␣appelez-vous␣?');
   readln(s);
   writeln('Bonjour␣',s);
   writeln('Votre␣nom␣comprend␣',length(s),'␣lettres.');
end.
```

- la première instruction de ce programme affiche à l'écran la chaîne `'Je␣m''appelle␣'` (notez la double apostrophe entre le `m` et le `a`), puis la constante `CHAINE1`, puis la constante `CHAINE2` et enfin un point final. L'affichage obtenu est
  `Je m'appelle TIMOLEON.`
  (notez l'absence d'apostrophes autour des différentes chaînes affichées)
- La dernière instruction utilise la fonction prédéfinie **length** qui donne la longueur de la chaîne passée en paramètre.

Remarque : On peut affecter un caractère à une chaîne de caractères, mais pas le contraire.

```
var
  s : STRING;
  c : CHAR;
begin
  c := 'a';
  s := c; // affectation acceptée
  s := 'b';
  c := s; // affectation non acceptée
end.
```

2.2 Notation indicielle

Une chaîne de caractères est, par définition, composée de caractères. Ces caractères sont *indexés* à partir de 1. Le premier caractère a pour indice 1, le second a pour indice 2, etc ... Le dernier caractère a pour indice la longueur de la chaîne.

On accède au caractère d'indice i d'une chaîne s avec la notation $s[i]$.

Si s est une chaîne, alors $s[i]$ est un caractère.

```
var
   s : STRING;
   c : CHAR;
begin
   s := 'TIMOLEON';
   c := s[3];
   {c = M}
end.
```

On peut modifier un caractère d'une chaîne de caractères à l'aide de la notation indicielle.

```
var
   s : STRING;
begin
   s := 'TIMULEON';
   s[4] := 'O';
   {s = TIMOLEON}
end.
```

```
// ecrireChaine(s) écrit à l'écran la chaîne s
// cette procédure est équivalente à write(s)
procedure ecrireChaine(const s : STRING);
var
  i : CARDINAL;
begin
  for i := 1 to length(s) do begin
     write(s[i]);
  end {for};
end {ecrireChaine};
```

Attention : l'accès à un caractère d'une chaîne s n'est possible que si l'indice i est compris entre 1 et la longueur de s. Toute tentative d'accès à un caractère d'indice strictement plus grand que la longueur entraîne une erreur à l'exécution signalée par `Runtime error 201`.

2.3 Concaténation

La *concaténation* de deux chaînes de caractères s_1 et s_2 est l'opération consistant à mettre bout à bout ces deux chaînes. La chaîne ainsi obtenue est appelée *concaténée* de s_1 et s_2.

En Pascal, c'est l'opérateur + qui permet d'exprimer la concaténation de deux chaînes de caractères. Par exemple, après l'instruction

```
s := CHAINE1+CHAINE2 ;
```

la variable `s` a pour valeur `TIMOLEON`.

La longueur de la chaîne de caractères $s1 + s2$ est égale à la somme des longueurs de chacune des deux chaînes.

2.4 Comparaison de chaînes

Les opérateurs =, <, <=, > et >= permettent de comparer deux chaîne de caractères.

Si `s1` et `s2` sont deux expressions de type **`STRING`**, alors

1. `s1 = s2` est vrai si et seulement si les deux chaînes de caractères `s1` et `s2` sont égales ;
2. `s1 < s2` est vrai si et seulement si la chaîne de caractère `s1` vient strictement avant `s2` dans l'ordre lexicographique ;

3. `s1 <= s2` est vrai si et seulement si la chaîne de caractère `s1` vient avant `s2` dans l'ordre lexicographique ou est égale à `s2` ;
4. etc . . .

3 Quelques fonctions prédéfinies sur les chaînes

Hormis la première, les fonctions présentées ici ne sont pas à connaître par cœur.

Longueur d'une chaîne `function length(s : STRING) : CARDINAL;`

`length`(s) donne la longueur de la chaîne s.
Exemples :
– **`length`**`('TIMOLEON')` = 8.
– **`length`**`('')` = 0.

`function copy(s : STRING; d, l : CARDINAL) : STRING;`

`copy`(s,d,l) donne la sous-chaîne de longueur l de la chaîne s débutant à l'indice d.
Exemples :

Extraction d'une sous-chaîne – **`copy`**`('TIMOLEON',3,4)` = `MOLE`.
– **`copy`**`('TIMOLEON',1,8)` = `TIMOLEON`.

Position d'une chaîne dans une autre `function pos(s1,s2 : STRING) : CARDINAL;`

`pos`(s_1,s_2) donne l'indice de la position de la chaîne s_1 dans la chaîne s_2 si elle s'y trouve, et donne 0 sinon.
Exemples :
– **`pos`**`('MOLE','TIMOLEON')` = 3.
– **`pos`**`('MULE','TIMOLEON')` = 0.

4 Exercices

Exercice 1. Sur le modèle de la procédure `ecrireChaine`, réalisez une procédure nommée `ecrireVertical` qui écrit verticalement la chaîne passée en paramètre. Par exemple, pour la chaîne `vertical` la procédure écrira à l'écran

```
v
e
r
t
i
c
a
l
```

Exercice 2. Sur le modèle de la procédure `ecrireChaine`, réalisez une procédure nommée `ecrireChaineAlEnvers` qui écrit à l'envers la chaîne passée en paramètre. Par exemple, pour la chaîne `repus` la procédure écrira à l'écran `super`.

Exercice 3. *Sous-chaînes*
Une *sous-chaîne* d'une chaîne de caractères s est une chaîne composée de caractères consécutifs de s. Elles peuvent être caractérisées par l'indice de début et leur longueur.
Voici par exemple quelques sous-chaînes pour la chaîne s = `TIMOLEON` :

Sous-chaîne	début	longueur
`T`	1	1
`IMO`	2	3
`LEON`	5	4
`TIMOLEON`	1	8

Question 1.
Réalisez la fonction

```
// sousChaine(s,debut,long) = sous-chaine de s de longueur long,
//                 débutant à l'indice debut
// CU : debut + long − 1 ≤ length(s)
function sousChaine(s : STRING; debut, long : CARDINAL) : STRING;
```

Remarque : la fonction `sousChaine` est prédéfinie en PASCAL et se nomme **copy**.

Question 2.
Réalisez la fonction

```
// positionSousChaine(s1,s2) = donne la position de s1 dans s2
//                 si s1 est dans s2
//                                   donne 0 sinon
function positionSousChaine(s1,s2 : STRING) : CARDINAL;
```

Remarque : la fonction `positionSousChaine` est prédéfinie en PASCAL et se nomme **pos**.

Exercice 4. *Palindromes*

Un *palindrome* est un mot qui se lit de la même façon de gauche à droite et de droite à gauche. `ICI`, `ETE`, `RADAR` sont des palindromes.

Réalisez un prédicat qui est vrai si et seulement si la chaîne passée en paramètre est un palindrome.

Exercice 5. *Initialisations de tas de cartes*

Les descriptions des tas de cartes sont des chaînes de caractères.

Question 1. Initialisez le tas 1 avec une description de tas donnée par l'utilisateur.

Question 2. Initialisez le tas 1 avec des trèfles dont le nombre est donné par l'utilisateur.

Question 3. Même question mais chaque carte étant de couleur quelconque.

Exercice 6. *Initiale et autres*

Question 1. Réalisez une fonction `initiale` qui donne le caractère initial (c'est-à-dire le premier) d'une chaîne de caractères. Cette fonction a-t-elle des contraintes d'utilisation ?

Question 2. Réalisez une fonction `finale` qui donne le caractère final (c'est-à-dire le dernier) d'une chaîne de caractères. Cette fonction a-t-elle des contraintes d'utilisation ?

Question 3. Réalisez une fonction `saufInitiale` qui donne la chaîne passée en paramètre sans son initiale. Cette fonction a-t-elle des contraintes d'utilisation ?

Exercice 7. *Préfixe*

Une chaîne de caractères s_1 est un *préfixe* d'une autre s_2 si la chaîne s_1 est le début de la chaîne s_2. À titre d'exemple, les préfixes de `'TIMOLEON'` sont '', `'T'`, `'TI'`, `'TIM'`, `'TIMO'`,`'TIMOL'`, `'TIMOLE'`, `'TIMOLEO'` et `'TIMOLEON'`.

Réalisez de dexu façons différentes une fonction qui teste si une chaîne (premier paramètre de la fonction) est un préfixe d'une autre (second paramètre de la fonction).

Exercice 8. Réalisez une fonction qui retourne le plus petit (aus sens de l'ordre défini dans la section 1.3) caractère de la chaîne de caractères passée en paramètre.