**Initiation à la programmation**

# TP : Chaînes de caractères

**Objectifs du TP** Ce TP a pour but de vous faire manipuler des chaînes da caractères.

## 1 Caractères

**Question 1.** Réalisez un programme qui affiche tous les caractères dont les codes ASCII sont compris entre 32 et 127.

Chaque caractère devra être affiché précédé de son code.

Voici un extrait de l'affichage souhaité :

```
...
 42: *   43: +   44: ,   45: -   46: .   47: /   48: 0   49: 1   50: 2    51: 3
 52: 4   53: 5   54: 6   55: 7   56: 8   57: 9   58: :   59: ;   60: <    61: =
...
```

On peut indiquer aux instructions d'affichage le format souhaité c'est à dire le nombre de caractères qu'on veut utiliser à l'aide de la syntaxe suivante :

```
var a:CARDINAL;
...
begin
  a := 2;
  write(a:15); // a sera affiché sur 15 caractères.
end;
```

### 1.1 Caractères et ordre

Les caractères sont rangés dans l'ordre de leur code ASCII. Il est tout à fait possible d'écrire l'expression (booléenne) suivante :

```
'a'<'B'
```

Cette expression vaut **false** car le code ASCII du a minuscule est supérieur au code ASCII du B majuscule (cf ??).

**Question 2.** En vous inspirant du code suivant réaliser une fonction nommée `est_minuscule`

```
function est_majuscule(const c: CHAR):BOOLEAN;
begin
   est_majuscule:= (c>='A' ) and (c<='Z');
end (*est_majuscule*)
```

### 1.2 Étoiles

**Question 3.** Réalisez la fonction `etoile` dont les spécifications sont les suivantes :

```
// la fonction étoile fabrique une chaine
// de n étoiles.
function etoile(const n: CARDINAL):STRING;
```

### 1.3 Triangle rectangle

**Question 4.** Réalisez un programme qui affiche la figure suivante :

```
*
**
***
****
*****
```

Bien entendu le nombre de lignes est donné par l'utilisateur du programme.

## 1.4 Triangle isocèle

**Question 5.** Réalisez un programme qui affiche la figure suivante :

```
    *
   ***
  *****
 *******
*********
```

## 1.5 Encadrer

**Question 6.** Réalisez la procédure `encadrer` dont les spécification sont les suivantes :

```
// la procédure encadrer affiche la
// chaine s encadrée par des étoiles.
procedure encadrer(const s:STRING);
```

Par exemple avec la chaîne s=`'timoleon'`, la procédure `encadrer` affiche

```
**********
*timoleon*
**********
```

# 2 Triangle de Sierpinski

Voici les 31 lignes produites par l'exécution d'un programme :

```
*
**
* *
****
*   *
**  **
* * * *
********
*       *
**      **
* *     * *
****    ****
*   *   *   *
**  **  **  **
* * * * * * * *
****************
*               *
**              **
* *             * *
****            ****
*   *           *   *
**  **          **  **
* * * *         * * * *
********        ********
*       *       *       *
**      **      **      **
* *     * *     * *     * *
****    ****    ****    ****
*   *   *   *   *   *   *   *
**  **  **  **  **  **  **  **
* * * * * * * * * * * * * * * *
```

et voici le corps de ce programme

```
const MAX = 30;
var
  s : STRING;
  i : CARDINAL;
begin
```

```
  s := '*';
  writeln(s);
  for i := 1 to MAX do begin
    s := transforme (s);
    writeln(s);
  end {for};
end.
```

Comme le montre ce programme chacune des lignes affichées l'a été par la seule instruction **writeln(s);**. Les valeurs successives de la chaîne `s`, initialisée à `'*'`, sont obtenues par une transformation programmée dans la fonction `transforme`. Chaque ligne se déduit donc de la ligne qui la précède par une règle simple.

**Question 7.** Récupéréz le programme exécutable `sierpinski` (valable uniquement pour environnement Linux) et lancez son exécution pour différentes valeurs du nombre de lignes souhaitées. Si nécessaire, changez les droits sur le fichier `sierpinski` avec la commande

```
chmod 744 sierpinski
```

à taper dans un terminal.

**Question 8.** Étudiez attentivement l'affichage pour comprendre en quoi consiste la transformation.

**Question 9.** Programmez ensuite la fonction `transforme`.

# 3 Une chanson connue...

Voici la chanson dont vous allez « programmer » le texte.

Buvons un coup ma serpette est perdue
Mais le manche, mais le manche
Buvons un coup ma serpette est perdue
Mais le manche est revenu

Bavas a ka ma sarpatta a parda
Ma la macha, ma la macha
Bavas a ca ma sarpatta a parda
Ma la macha a ravana

Beves e ke me serpette e perde
Me le meche, me le meche
Beves e ke me serpette e perde
Me le meche e revene

...

Chaque couplet qui suit le premier est une transformation de celui-ci en remplaçant toutes les voyelles par une autre voyelle fixée.

## 3.1 `est_voyelle`

**Question 10.** Réalisez une fonction paramétrée par un caractère dont le résultat est un booléen, qui est vrai lorsque le caractère est une voyelle.

## 3.2 Un couplet

**Question 11.** Réalisez une fonction nommée `bavazaka` paramétrée par une chaîne `s` et un caractère `c`, dont le résultat est une nouvelle chaîne obtenue en remplaçant toutes les voyelles de la chaîne `s` par le caractére `c` .

Exemple :

```
writeln(bavazaka('ma serpette est perdue','a'));
// affiche:
// ma sarpatta ast pardaa
```

## 3.3 La chanson

**Question 12.** Utilisez vos fonctions pour écrire un programme qui affiche le texte complet de la chanson avec un couplet par voyelle.

Remarque : en fait une chanson ayant vocation à être écoutée et non lue, la transformation ne doit transformer que les voyelles oralisées. Par exemple, le mot `coup` est transformé avec la voyelle a en le mot prononcé `ka`. Votre

programme ne transformera pas les sons mais le texte. Ainsi le mot `coup` sera transformé en `caap`, et le mot `buvons` en le mot `bavans`

Vous définirez le premier couplet comme une constante de type chaîne de carcatères dans votre programme de la façon suivante :

```
const
  // pour Windows, remplacer char(10) par char(13)+char(10)
  PASSAGEALALIGNE = char(10);

  couplet1 = 'Buvons␣un␣coup␣ma␣serpette␣est␣perdue'+PASSAGEALALIGNE+
     'Mais␣le␣manche,␣mais␣le␣manche'+PASSAGEALALIGNE+
     'Buvons␣un␣coup␣ma␣serpette␣est␣perdue'+PASSAGEALALIGNE+
     'Mais␣le␣manche␣est␣revenu';
```