Les string et Char

A- Une variable 'String' peut contenir une chaîne de caractères, des données alphanumériques.

Pour travailler avec les 'String', on peut:

1-Utiliser les membres de la Classe String.

2-Utiliser les méthodes du VisualBasic 3-Exemple

4-Comparaison de caractères

5- Unicode

B -Une variable 'Char' peut contenir un caractère.

A- STRING.

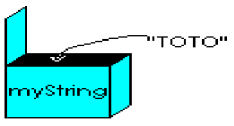
Il faut **déclarer** une variable avant de l'utiliser, pour cela on utilise l'instruction DIM

```
DIM STR As String
```

'Déclare une variable nommée **Str** et qui peut contenir une chaîne de caractère.

Cette variable peut être utilisée pour conserver une chaîne de caractère.

```
STR= "TOTO"
```




```
'On met la chaîne de caractères "TOTO" dans la variable STR
```

Remarquons que pour définir une chaîne de caractères il faut utiliser des " " : Ce qui est entre " et " est la chaîne de caractères. On parle de **chaîne littérale: (**une représentation textuelle d'une valeur particulière)

Après avoir été créée, une String contient 'Nothing' c'est à dire rien (même pas une chaîne vide: ""); il faudra l'initialiser pour qu'elle contienne quelque chose.

```
DIM str As String    'str contient Nothing
(pas le texte "Nothing"!! cela signifie qu'elle ne pointe sur 'rien'
str= ""              'str contient "" : chaîne vide de longueur 0
str= "TOTO"          'str contient "TOTO"
```

Notez bien l'importance des guillemets:

```
A est la variable A
"A" est une chaîne de caractères contenant le caractère "A"
Exemple:
Dim A As String= "Visual"
Dim B As String= "Basic"
Label.text = "A+B"  affiche bêtement la chaîne  « A+B »
Label.text =  A+B    affiche "VisualBasic" 'on affiche les variables.
```

Notez enfin que " ", l'**espace** est un caractère à part entière.

On peut **initialiser** la variable en même temps qu'on la déclare.

```
Dim Chaine as string = "Toto
```

On peut déclarer plusieurs variables d'un même type sur une même ligne.

```
Dim x, y, z As String    'Déclare 3 variables 'String'
```

On utilise GetType pour connaître le type d'une variable.

```
x.GetType.ToString
y.GetType.ToString
z.GetType.ToString
```

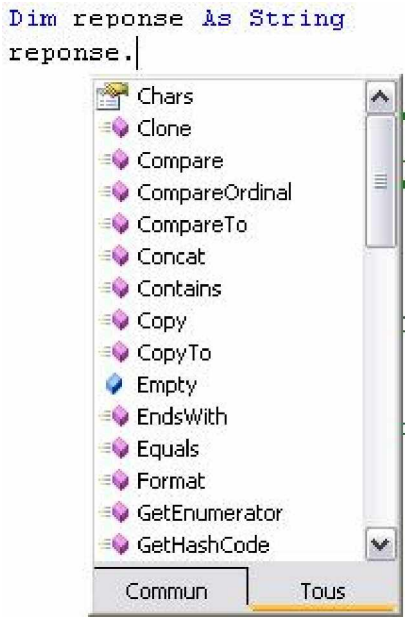
donne

```
System.String
System.String
System.String
Ce qui prouve que les 3 variables sont bien des Strings. (ce qui n'était pas le cas en VB6)
```

1- System.String est une Classe du Framework:

Le type **System.String** ou **String** (Chaîne de caractères) est une **Classe** qui a des méthodes.

Pas besoin de connaître toutes les méthodes, il suffit (Après déclaration de la String par DIM réponse AS String) de taper "." .Et vous voyez apparaître toutes les propriétés et méthodes :



Par exemple la méthode **.ToUpper**

Mettre en majuscules une chaîne de caractère

```
str=str.ToUpper()
```

Si str contenait "**abc**" il contiendra "**ABC**"

.ToLower transforme par contre la chaîne en minuscule.

.Trim

Permet de supprimer des caractères en début et fin de chaîne.

```
Dim a As String = "#@Informatique@#"

Dim b As Char() = {"#", "@"}    'b est un tableau de Char contenant les caractères à supprimer.

a=a.Trim(b)  Donne a= "Informatique"
```

Attention : Bien utiliser Char() qui est un tableau de caractères pour définir les caractères à supprimer.

(Dim b As String= "#@" est déconseillé car produisant des résultats curieux.)

Pour enlever les espaces avant et après la chaîne (Cas le plus fréquent) :

```
a=" Bonjour "

a=a.Trim(" ")    'donne a="Bonjour"
```

Il existe aussi TrimStart et TrimEnd pour agir seulement sur le début ou la fin de la chaîne.

Length : Taille d'une chaîne en nombre de caractère.

Afficher la taille de la chaîne « VB »

```
Dim s As String= "VB"

MsgBox(s.Length.ToString) 'Affiche 2
```

Concat :

Concaténation de plusieurs chaînes : mise bout à bout :

```
s=String.Concat(a,b)
```

Il est plus rapide de faire : `s= a & b`

(`s=a+b` marche mais est déconseillé)

Insert :

Insère une chaîne dans une autre.

```
Dim s As String= "VisualBasic"

s= s.Insert(6,"  " ) 'Donne s= "Visual  Basic"
```

Noter: le premier caractère a la position 0.

Remove :

Enlève des caractères à une certaine position dans une chaîne.

```
Dim s As String= "VisualBasic"

s= s.Remove(2,7 ) 'Donne s= "Viic"
```

Replace :

Remplace dans une chaîne de départ, toutes les occurrences d'une chaîne par une autre.

```
Resultat=ChaineDépart.Replace(ChaîneARemplacer,ChaîneQuiRemplace)

Dim s As String= "Visual_Basic"

s= s.Replace("_"," " ) ‘Donne s= "Visual  Basic"
```

Autre exemple:

L'utilisateur a tapé une date, mais avec comme séparateur des ".", comme on le verra plus loin, il est nécessaire d'utiliser plutôt les "/", pour cela on utilise Replace

```
Dim ladate as string= "12.02.1990"

ladate= ladate.Replace("."."/" ) ‘Donne ladate= "12/02/1990"
```

Split :

Découpe en plusieurs sous chaînes une chaîne de départ, cela par rapport à un séparateur.

Exemple :

Je récupère dans un fichier une chaîne de mots ayant pour séparateur « ; », je veux mettre chaque mot dans un tableau.

```
‘Chaîne contenant les mots séparés par « ; »

Dim s As String= "Philippe;Jean ;Toto"

Dim separateur As Char = ";"

Dim nom() As String

nom=s.Split(separateur)
```

Donne :

```
      nom(0)= "Philippe"

      nom(1)= "Jean"

      nom(2)= "Toto"
```

Remarque: Quand on déclare le tableau nom(), on ne donne pas le nombre d'élément, c'est Split qui crée autant d'élément qu'il faut.

En Framework 2, on peut utiliser plusieurs séparateurs différend:

```
nom=s.Split( New Char() {" "c, ","c, "."c })
```

ici on a 3 séparateurs: l'espace, la virgule et le point.

le c après chaque séparateur veut dire Char, car les séparateurs sont des caractères.

On peut ajouter 2 paramètres permettant d'indiquer le nombre de ligne maximum et forcer l'élimination des lignes vides.

```
Dim sep() As Char={" "c, ","c, "."c}

Dim nom() As String = S.Split ( sep, 100,
StringSplitOptions.RemoveEmptyEntries )
```

.Join

Concatène tous les éléments d'un tableau et peut ajouter des séparateurs.

Si myLines() est un tableau de String , je veux ajouter ces lignes bout à bout en les séparant d'un retour à la ligne.

```
Dim myText As String = String.Join ( ControlChars.CrLf, myLines)
```

.IndexOf .LastIndexOf

Indique le numéro du caractère, la position (la première occurrence) ou une chaîne à chercher est trouvée dans une autre. Recherche en commençant par la fin avec LastIndexOf

```
Dim a As String= "LDF.EXE"

Dim r As Char()={"."}

a.IndexOf(r)  retourne 3
```


Se souvenir : le premier caractère est en position **0 en .Net**.

.LastIndexOf retourne la dernière occurrence.

.IndexOfAny .LastIndexOfAny (Framework 2)

Indique le numéro du caractère, la position (la première occurrence) ou une chaîne à chercher est trouvée dans une autre avec en plus possibilité d'indiqué la position de départ.

```
Dim a As String= "LDF.EXE"

Dim r As Char()={"."}

a.IndexOfAny(r)  recherche à partir du début de chaîne.

a.IndexOfAny(r,2)  recherche à partir du deuxième caractère.
```

Autre exemple: On recherche ici plusieurs caractères (en fait un tableau de Char)

```
Dim str As String ="gfdjzak;,vdqsygeak"
```

```
Dim start As Integer =2

Dim at As Integer

Dim count As Integer =5

Dim target As String = "ou" 'chaîne à chercher

Dim anyOf As Char() = target.ToCharArray() 'on transforme la chaîne en
tableau de char

at = str.IndexOfAny(anyOf, start, count) 'on cherche le tableau de Char
anyOf dans str à partir de la position start et sur count caractères.
```

.Compare

Compare 2 chaînes :

```
Dim rep As Integer

rep=String.Compare(a,b)
```

Retourne un entier

Négatif si a<b

0 si a=b

Positif si a>b

On peut comparer des sous chaînes et indiquer la sensibilité à la casse (Framework 2):

```
rep= String.Compare(a, 2, b, 2, 10, True)
```

Ici on compare 10 caractères en commençant par le deuxième caractère de chaque chaîne en mode sensible à la casse (majuscules<>minuscule).

.Substring

Extrait une partie d'une chaîne.

Le premier paramètre indique la position de départ; le second, le nombre de caractères à extraire.

```
Dim a As String= "Informatique"

MessageBox.show(a.Substring(2,3)) 'Affiche  for
```

Le premier paramètre indique la position du caractère où doit commencer la sous-chaîne, en commençant à la position 0. (les caractères sont comptés 0, 1, 2, 3....

Le second paramètre la longueur de la sous-chaîne.

Exercice 1: comment obtenir les 4 caractères de droite:

```
Dim a As String= "Informatique"

MessageBox.show(a.Substring(A.Length-4)) ‘Affiche  ique
```

Ici on omet le second paramètre,la longueur de la sous-chaîne, va jusqu'a la fin de la chaîne.

Exercice 2: comment obtenir les 3 caractères de gauche:

```
Dim a As String= "Informatique"

MessageBox.show(a.Substring(0, 3)) ‘Affiche  inf
```

.Chars

Une chaîne peut être perçue comme un **tableau de caractères** (instances Char) ; vous pouvez extraire un caractère particulier en faisant référence à l'index de ce caractère par l'intermédiaire de la propriété Chars. Par exemple :

```
Dim maString As String = "ABCDE"
Dim monChar As Char
monChar = maString.Chars(3) ' monChar = "D"
```

On peut créer des chaînes avec la Classe String:

```
myString = New String(" ", 15)    'Créer une chaîne de 15 espaces
```

.PadRight

Aligne les caractères de cette chaîne à gauche et remplit à droite en ajoutant un caractère Unicode spécifié pour une longueur totale spécifiée.

```
Dim str As String
Dim pad As Char
str = "Nom"
pad = Convert.ToChar(".")
Console.WriteLine(str.PadRight(15, pad)) ' Affiche Nom.....
```

PadLeft fait l'inverse.

.StartsWith() et EndsWith()

Permettent de tester si une string commence ou se termine par une string, retourne True ou False.

Tester si la String s commence par "abc" et se termine par "xyz":

```
If s.StartWith ("abc") And s.EndWith ("xyz") then
```

2- On peut aussi utiliser les instructions 'Visual Basic':

Si vous débuter, laisser de coté ces instructions Visual Basic: elle font double emploi avec la classe String, elle ne sont pas toujours cohérentes avec le reste et cela embrouille.

Utiliser uniquement la classe String.

Elles sont bien connues des 'anciens' et font partie intégrante de VisualBasic) et sont parfois plus simples. Mais elles ne fonctionnent pas comme des Objets mais comme des instructions.

Elle font partie de l'espace de nom Microsoft.VisualBasic, il est 'chargé' par défaut et il n'y a pas lieu de l'importer. Par contre quand certains 'mots' sont communs à plusieurs classes ou instructions, il peut y a avoir ambiguïté et il faut utiliser dans ce cas la syntaxe complète. Cela semble le cas pour left qui est un mot clé Vb mais aussi une propriété des contrôles. Pour lever l'ambiguïté il faut écrire `Microsoft.VisualBasic.left(C,i)` par exemple.

Ces méthodes font souvent double emploi avec les méthodes de la classe String:


Attention : le premier caractère est en position **1 dans les instructions VB**.

Mid:

Permet de récupérer une sous-chaîne.

```
MaString = "Mid Demonstration"
a = Mid(MaString, 1, 3) ' Retourne "Mid".
```

Retourne 3 caractères à partir **du premier**

Le premier paramètre indique la position du caractère où doit commencer la sous-chaîne, en commençant à la position 1. (les caractères sont comptés 1, 2, 3...; on rappelle qu'avec SubString la sous-chaîne, commence à la position 0.

```
a = Mid(MaString, 14) ' Retourne "tion": du 14ème à la fin (pas de 3ème argument)
```

Mid permet aussi de remplacer une string dans une string

```
Mid(MaString, 1, 3) = "Fin"  => MaString="Fin Demonstration"
```

Left, Right (Pas d'équivalent dans le Framework)

Retourne x caractères de gauche ou de droite:

```
a=Right(MaString,2)    'a="on"
```

```
a=Microsoft.VisualBasic.Left(MaString,2)    'a="Mi"
```

Notez bien que, pour lever toute ambiguïté avec les méthodes left d'autres classes, il faut indiquer Microsoft.VisualBasic.Left.

Len.

Retourne la longueur de la chaîne:

```
MyLen = Len(MaString) ' Retourne 17.
```

LTrim, RTrim

Enlève les espaces à gauche ou à droite d'une chaîne.

```
a=LTrim("  RRRR")  ' a="RRR"
```

InStr

Retourne un entier spécifiant la position de début de la première chaîne à l'intérieur d'une autre.

```
n=InStr(1,"aaaaRaa","R")        'retourne 5
```

Recherche à partir du premier caractère, à quelle position se trouve 'R' dans la chaîne "aaaaRaa"

Si la chaîne n'est pas trouvée , retourne 0

InStrRev

Recherche aussi une chaîne mais de droite à gauche. la position de départ est le 3ème argument.

```
InStrRev (Ch1, Ch2 , PosDépart)
```

StrComp Compare 2 chaînes.

Space

Retourne une chaîne d'espace: Space(10) retourne "   "

StrDup

Retourne une chaîne de caractères par duplication d'un caractère dont on a spécifié le nombre.

```
maString = StrDup(5, "P") ' Retourne "PPPPP"
```

Asc

Retourne le code de caractère du caractère entré. Il peut être compris entre 0 et 255 pour les valeurs du jeu de caractères codé sur un octet (SBCS) et entre -32 768 et 32 767 pour les valeurs du jeu de caractères codé sur deux octets (DBCS). La valeur retournée dépend de la page de codes

AscW retourne le code Unicode du caractère entré. Il peut être compris entre 0 et 65 535.

```
x=Asc("A")    'retourne 65

x=Asc("ABCD") 'retourne 65 : Seul le premier caractère est pris en compte
```

Chr et ChrW

Retourne le caractère associé au code de caractère.

```
Chr(65)   retourne "A"    'cela dépend de la page de code.
```

On peut donner le numéro du caractère en hexadécimal, dans ce cas on le fait précéder de &H

```
Chr(&H20)   est équivalent de Chr(32) et retourne un caractère " ".

ChrW  retourne le caractère correspondant à l'Unicode
```

GetChar

Retourne le caractère d'une chaîne à une position donnée.

```
Dim maString As String = "AIDE"
Dim monChar As Char
monChar = GetChar(maString, 3) ' monChar = "D"
```

LCase Ucase

Retourne la chaîne en minuscule ou majuscule:

```
Lowercase = LCase(UpperCase)
```

Lset Rset

Retourne une chaîne alignée à gauche avec un nombre de caractère.

```
Dim maString As String = "gauche"
Dim r As String
r = LSet(maString, 2) ' Retourne "ga"
```

Si la chaîne de départ est plus courte que la longueur spécifiée, des espaces sont ajoutés.

```
r = LSet(maString, 8) ' Retourne "gauche  "
```

StrRevers

Retourne une chaîne ou les caractères ont été inversés:

```
Dim maString As String = "STRESSED"

Dim revString As String

revString = StrReverse(myString)    ' Retourne "DESSERTS"
```

Like:

Instruction **hyper puissante**: Like, elle **compare** une chaîne String avec un modèle (Pattern), elle permet de voir si la chaîne contient ou ne contient pas un ou des caractères, ou une plage de caractères. (c'est l'équivalent des **expressions régulières du Framework**)

```
result = String Like Pattern
```

Si *string* correspond à *pattern*, la valeur de *result* est **True** ; s'il n'y a aucune correspondance, la valeur de *result* est **False**. Si string et pattern sont une chaîne vide, le résultat est **True**. Sinon, si *string* ou *pattern* est une chaîne vide, le résultat est **False**.

L'intérêt de Like est que l'on peut y mettre des **caractères génériques:**

? veut dire tout caractère unique

***** veut dire * ou plusieurs caractères.

veut dire tout chiffre.

[caractères] veut dire tout caractères présent dans la liste.

[!caractères] veut dire tout caractères NON présent dans la liste.

- trait d'union permet de spécifier un début et une fin de plage.

Exemple:

```
Dim R As Boolean
R = "D" Like "D" ' Est ce que "D" est égal à "D"? => True.


R = "F" Like "f" ' Est ce que "F" est égal à "f"? => False.


R = "F" Like "FFF" ' Est ce que "F" est égal à "FFF"? => False.


R = "cBBBc" Like "c*c" ' Est ce que "cBBBc" répond au pattern (avoir un "c"
au
'début, un  "c" à la fin, et des caractères au milieu? Retourne True.


R = "J" Like "[A-Z]" ' Est ce que "J" est contenu dans les caractères
allant de
'  A à Z? Retourne True.
```

```
R = "I" Like "[!A-Z]" ' Est ce que "I" n'est PAS dans les caractères allant de
'  A à Z? Retourne  False.


R = "a4a" Like "a#a" ' Est ce que "a4a" commence et finie par un
' "a" et à un nombre entre les 2? Retourne True.


R = "bM6f" Like "b[L-P]#[!c-e]" ' Est ce que "bM6f"
'commence par  "b",
'a des caractères entre L et P
'un nombre
'se termine par un caractère non compris entre c et e
'retourne True
```

3- Un exemple:

Combinaison de chaînes de caractères, de variables..

Souvent, on a besoin d'afficher une combinaison de chaînes littérales, le contenu de variables, des résultats de calcul; c'est possible.

Exemple :

Pour afficher dans un label le carré de X est X2, avec une valeur dans la variable x :

```
Dim X As Integer = 2

Label1.text= "Le carré de " & X & "  est " & X * X
```

Ce qui est entre guillemets est affiché tel quel. C'est le cas de "Le carré de " et de " est "

Ce qui n'est pas entre guillemets est évalué, le résultat est affiché. C'est le cas de X et X*X

Pour ne faire qu'une chaîne on ajoute les bouts de chaînes avec l'opérateur '&'.

Notez l'usage d'espace en fin de chaîne pour que les mots et les chiffres ne se touchent pas.

```
Dim X As Integer

X=2

Label1.text= "Le carré de " & X & "  est " & X * X
```

Affiche dans le label : « Le carré de 2 est 4 »

Voir des **exemples de code** dans ve1-1

4- Comparaison de caractères et 'Option Compare'

On peut comparer 2 String

```
Dim s1 As String ="ABCD"

Dim s2 As String ="XYZ"
```

Dans ce cas `s1<s2` est vraie.

Par défaut (Option Compare Binary)

Les caractères sont classés dans un ordre croissant (l'ordre de leur code unicode)

Voyons l'ordre des certains caractères particuliers:

" " +,-./ 0123456789 :;ABCDEF abcdef èéê

On constate que l'ordre est espace puis quelques caractères spéciaux, les chiffres, les majuscules puis les minuscules, les accentués.(voir le tableau d'unicode)

Ainsi `B<a`

En utilisant **Option Compare Binary**, la plage [A–E] correspond à *A*, *B*, *C*, *D* et *E*.

Avec Option Compare Text

Les caractères sont classées dans un ordre qui reflète plus la réalité d'un texte:

Toutes les types de a: *A*, *a*, À, à, puis tous les types de b: *B*, *b...*

Avec **Option Compare Text**, [A–E] correspond à *A*, *a*, À, à, *B*, *b*, *C*, *c*, *D*, *d*, *E* et *e*. La plage ne correspond pas à Ê ou ê parce que les caractères accentués viennent après les caractères non accentués dans l'ordre de tri.

Ainsi `B>a`

L'ordre des caractères est donc défini par **Option Compare** et aussi les paramètres régionaux du système sur lequel s'exécute le code.

Grande règles de comparaison:

La comparaison s'effectue de gauche à droite.

La comparaison s'effectue sur le premier caractère de chaque chaîne.

Si le premier caractère est identique, la comparaison se fait sur le deuxième caractère...

```
"zz" > "za"  est vrai
```

En cas de chaîne du type "zz" et "zzz" , la seconde est supérieure

```
"zz" < "zzz"  est vrai
```

Il y a quelques **pièges**:

Si je veux créer des chaînes du genre 'un nombre puis le mot string' et qu'elles soient classées dans un ordre logique pour l'humain.

Je vais taper: "1string", "2string", "10string", "11string", "100string"

Le classement par Vb sera 'surprenant' car les chaînes seront classées dans cet ordre:

```
"100string", "10string", "11string", "1string","2string"
```

Pourquoi? c'est l'application stricte des règles de comparaison: regardons le troisième caractère des 2 premières chaînes (les 2 premiers caractères étant égaux), "0" est bien inférieur à "s" donc "100string" < "10string" est vrai!!

Pour résoudre le problème et obtenir un classement correct, il faut écrire des blocs numériques de même longueur et alignés à droite:

Écrire 010string et non 10string.

```
   "001string", "002string", "010string", "011string", "100string" ' ici le trie est dans le bon ordre.
```

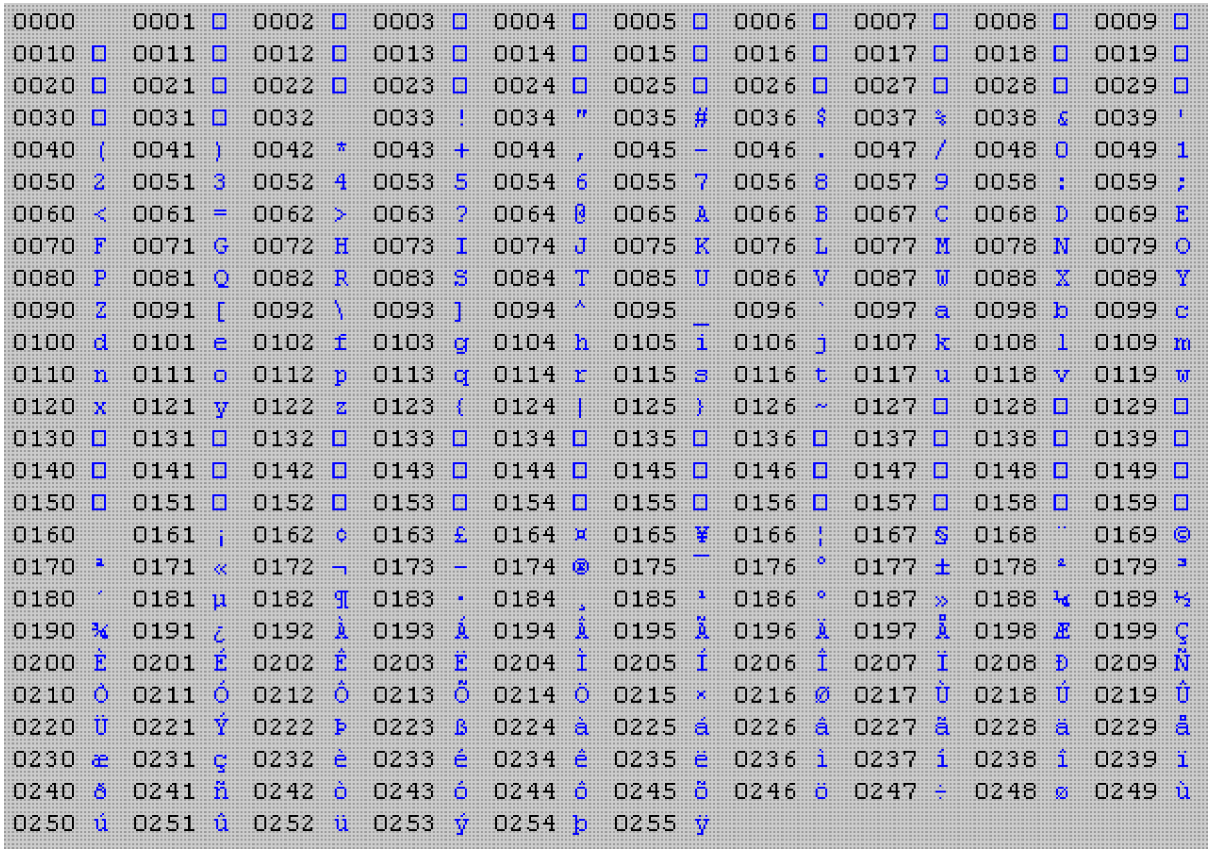
5- Unicode:

Les variables **string** sont stockées sous la forme de séquences de 16 bits (2 octets) non signés dont les valeurs sont comprises entre 0 et 65 535. Chaque nombre représente un caractère **Unicode**. Une chaîne peut contenir jusqu'à 2 milliards de caractères.

L'unicode est donc un codage de caractères sur 16 bits qui contient tous les caractères d'usage courant dans les langues principales du monde.

Les premiers 128 codes (0–127) Unicode correspondent aux lettres et aux symboles du clavier américain standard. Ce sont les mêmes que ceux définis par le jeu de caractères ASCII (ancien codage sur un octet). **Les 128 codes suivants** (128–255) représentent les caractères spéciaux, tels que les lettres de l'alphabet latin, les accents, les symboles monétaires et les fractions. Les codes restants sont utilisés pour des symboles, y compris les caractères textuels mondiaux, les signes diacritiques, ainsi que les symboles mathématiques et techniques.

Voici les 255 premiers



0000	0001 □	0002 □	0003 □	0004 □	0005 □	0006 □	0007 □	0008 □	0009 □
0010 □	0011 □	0012 □	0013 □	0014 □	0015 □	0016 □	0017 □	0018 □	0019 □
0020 □	0021 □	0022 □	0023 □	0024 □	0025 □	0026 □	0027 □	0028 □	0029 □
0030 □	0031 □	0032	0033 !	0034 "	0035 #	0036 $	0037 %	0038 &	0039 '
0040 (	0041)	0042 *	0043 +	0044 ,	0045 -	0046 .	0047 /	0048 0	0049 1
0050 2	0051 3	0052 4	0053 5	0054 6	0055 7	0056 8	0057 9	0058 :	0059 ;
0060 <	0061 =	0062 >	0063 ?	0064 @	0065 A	0066 B	0067 C	0068 D	0069 E
0070 F	0071 G	0072 H	0073 I	0074 J	0075 K	0076 L	0077 M	0078 N	0079 O
0080 P	0081 Q	0082 R	0083 S	0084 T	0085 U	0086 V	0087 W	0088 X	0089 Y
0090 Z	0091 [	0092 \	0093]	0094 ^	0095 _	0096 `	0097 a	0098 b	0099 c
0100 d	0101 e	0102 f	0103 g	0104 h	0105 i	0106 j	0107 k	0108 l	0109 m
0110 n	0111 o	0112 p	0113 q	0114 r	0115 s	0116 t	0117 u	0118 v	0119 w
0120 x	0121 y	0122 z	0123 {	0124 |	0125 }	0126 ~	0127 □	0128 □	0129 □
0130 □	0131 □	0132 □	0133 □	0134 □	0135 □	0136 □	0137 □	0138 □	0139 □
0140 □	0141 □	0142 □	0143 □	0144 □	0145 □	0146 □	0147 □	0148 □	0149 □
0150 □	0151 □	0152 □	0153 □	0154 □	0155 □	0156 □	0157 □	0158 □	0159 □
0160	0161 ¡	0162 ¢	0163 £	0164 ¤	0165 ¥	0166 ¦	0167 §	0168 ¨	0169 ©
0170 ª	0171 «	0172 ¬	0173 -	0174 ®	0175 ¯	0176 °	0177 ±	0178 ²	0179 ³
0180 ´	0181 µ	0182 ¶	0183 ·	0184 ¸	0185 ¹	0186 º	0187 »	0188 ¼	0189 ½
0190 ¾	0191 ¿	0192 À	0193 Á	0194 Â	0195 Ã	0196 Ä	0197 Å	0198 Æ	0199 Ç
0200 È	0201 É	0202 Ê	0203 Ë	0204 Ì	0205 Í	0206 Î	0207 Ï	0208 Ð	0209 Ñ
0210 Ò	0211 Ó	0212 Ô	0213 Õ	0214 Ö	0215 ×	0216 Ø	0217 Ù	0218 Ú	0219 Û
0220 Ü	0221 Ý	0222 Þ	0223 ß	0224 à	0225 á	0226 â	0227 ã	0228 ä	0229 å
0230 æ	0231 ç	0232 è	0233 é	0234 ê	0235 ë	0236 ì	0237 í	0238 î	0239 ï
0240 ð	0241 ñ	0242 ò	0243 ó	0244 ô	0245 õ	0246 ö	0247 ÷	0248 ø	0249 ù
0250 ú	0251 û	0252 ü	0253 ý	0254 þ	0255 ÿ				

Le petit carré indique un caractère non imprimable (non affichable), certains caractères sont des caractères de contrôle comme le numéro 9 qui correspondant à tabulation, le numéro 13 qui correspond au retour à la ligne..

B- CHAR.

Les variables **Char** sont stockées sous la forme de nombres 16 bits (2 octets) non signés dont les valeurs sont comprises entre 0 et 65 535. Chaque nombre représente un seul caractère **Unicode**. Pour les conversions entre le type **Char** et les types numériques il y a les fonctions **AscW** et **ChrW** qui peuvent être utilisées..

L'ajout du caractère de type littéral **C** à un littéral de chaîne force ce dernier a être un type **Char**. A utiliser surtout si **Option Strict (qui force à être strict..)** est activé.

Exemple:

```
Option Strict On
' ...
Dim C As Char
C = "A"c
```

String.ToCharArray:
Permet de passer une string dans un tableau de Char:

```
Dim maString As String = "abcdefghijklmnop"
Dim maArray As Char() = maString.ToCharArray
```

La variable maArray contient à présent un tableau composé de **Char**, chacun représentant un caractère de maString.

Pour mettre le tableau de Char dans une String:

```
Dim maNewString As String (maArray)
```

String.Chars():
vous pouvez extraire un caractère particulier en faisant référence à l'index de ce caractère par l'intermédiaire de la propriété Chars. Par exemple :

```
Dim maString As String = "ABCDE"

Dim monChar As Char

monChar = maString.Chars(3) ' monChar = "D"
```

Un caractère est-il numérique? un chiffre? une lettre? un séparateur? un espace?....

```
        Dim chA As Char
        chA = "A"c
        Dim ch1 As Char
        ch1 = "1"c
        Dim str As String
        str = "test string"

        Console.WriteLine(chA.CompareTo("B"c))          ' Output: "-1" '  A
est plus petit que B
        Console.WriteLine(chA.Equals("A"c))             ' Output: "True" '
Egal?
        Console.WriteLine(Char.GetNumericValue(ch1))    ' Output: 1
'Convertir en valeur numérique (double)
        Console.WriteLine(Char.IsControl(Chr(9)))       ' Output: "True"'
Est une caractère de contrôle?
        Console.WriteLine(Char.IsDigit(ch1))            ' Output: "True"'
Est un chiffre
        Console.WriteLine(Char.IsLetter(","c))          ' Output: "False"'
Est une lettre
        Console.WriteLine(Char.IsLower("u"c))           ' Output: "True" '
Est en minuscule
        Console.WriteLine(Char.IsNumber(ch1))           ' Output: "True" '
Est un nombre
        Console.WriteLine(Char.IsPunctuation("."c))     ' Output: "True" '
Est un caractère de ponctuation
        Console.WriteLine(Char.IsSeparator(str, 4))     ' Output: "True" '
Est un séparateur
        Console.WriteLine(Char.IsSymbol("+"c))          ' Output: "True" '
Est un symbole
        Console.WriteLine(Char.IsWhiteSpace(str, 4))    ' Output: "True" '
Est un espace
        Console.WriteLine(Char.ToLower("M"c))           ' Output: "m"        ' Passe en
```

Existe aussi IsLetterOrDigit, IsUpper.
Bien sur si on est en 'Option Strict' il faut ajouter .ToString à chaque ligne:

```
Console.WriteLine(Char.ToLower("M"c).ToString)
```

On note que l'on peut tester un caractère dans une chaîne:
```
Char.IsWhiteSpace(str, 4)
```

Autre manière de tester chaque caractères d'une String:
```
Dim V as string
For Each C As Char in V    'Pour chaque caractère de V..
      C...
Next
```
Ici la String est considérée comme une collection de Char. (C'est aussi une collection de String)
Mais on verra plus loin les collections et les boucles For Each.

Conversions Char <->Unicode
On rappelle que l'unicode est le mode de codage interne des caractères.
```
Dim monUnicode As Short = Convert.ToInt16 ("B"c) ' le code Unicode de B est 66.
Dim monChar As Char = Convert.ToChar (66) '   monChar="B"
```

Si vous souhaiter utiliser Asc et Chr de VisualBasic:
```
Dim monAscii As Short = Asc("B") 'donne le code Ascii ou l'Unicode (Ascw fait de même?)
Dim monChar As Char= Chr(66)
```

Voir des **exemples de code** dans ve1-1

Et les Chaînes de longueur fixe:

On a vu que les chaînes de longueur fixe n'existent pas en VB.NET (compatibilité avec les autres langages oblige), mais il y a moyen de contourner le problème:

On peut utiliser la **Classe de compatibilité VB6: à éviter**

(Il faut charger dans les références du projet Microsoft.VisualBasic.Compatibility et Compatibility Data)

```
Dim MaChaineFixe As New VB6.FixedLengthString(100)
```

Pour afficher la chaîne fixe utilisez MaChaineFixe.Tostring

Mais pour mettre une chaîne dans cette chaîne de longueur fixe!! galère!!!

`MaChaineFixe="ghg"` n'est pas accepté: on ne peut pas mettre une String dans une chaîne fixe

```
MaChaineFixe = CType("hg", Microsoft.VisualBasic.Compatibility.VB6.FixedLengthString) 'pas accepté non plus!!
```

Enfin ce type de chaîne fixe ne peut pas être utilisée dans les structures, mais il y a un autre moyen pour les structures. On verra cela plus loin.

Donc **les chaînes fixes sont à éviter.**