VB.NET : Le langage

1. Visual Basique ?
- insallation de visual basique 2008 Express
- Découverte de l'environnement
- projet en mode console
- Premier programme en VB

2. Découverte du langage et mise en pratique
- Variables
- Opérateurs arithmétiques
- Opérateurs de comparaison
- Branchements
- Boucles,
- Tableaux
- Fonctions
- Structures

1. Visual Basique ?

Insallation de visual basique 2008 Express

Découverte de l'environnement

Projet en mode console

Premier programme en VB

2. Découverte du langage et mise en pratique

Variables

<table>
<tr><th>VB</th><th>équivalence en C ?</th></tr>
<tr><td>

12 types de variables

```
TYPES       NB OCTETS       valeurs

Byte          1
SByte         1
Boolean       2             True-False
Char          2
Short         2
UShort        2
Integer       4
UInteger      4
Single        4
Object        4
Long          8
ULong         8
Double        8
Date          8
Decimal       16
String        2*taille
```

</td><td>

```
TYPES       NB OCTETS       valeurs

unsigned char      1
char               1
-NO-
unsigned short     2
short              2
unsigned short
int                4
unsigned int       4
float              4
pointeur           4
long               4
unsigned long      4
double             8
-NO-
-NO-
char*              1*taille
```

</td></tr>
<tr><td>

Déclaration des variables :

```
Dim <nom var> As <type var>

EX :
 Dim c As Char

R1 : toutes les variables sont
initialisées par défaut à 0 à la
déclaration.
```

</td><td>

```
<type> <nom var> <;>

EX :
  int res;

R1 : il n'y a pas d'initialisation
par défaut
```

</td></tr>
</table>

Portée des variables :	... identique
Bloc dans lequel elle est déclarée et sous blocs correspondants	
Concaténer 2 chaines :	**Différence**

Opérateur &
 	
Dim s1 As String="Bonjour"
Dim S2 As string="Monsieur"
Dim s3 As String = s1 & s2
	Obligation d'utiliser la fonction strcat() :
	char*s1="bonjour"
	char*s2="Monsieur"
	char s3[256];
	strcat(s3,s1);
	strcat(s3,s2);

Compatibilité des types entre eux :	**Différence**
Les opérations : le type de l'expression est celui du résultat L'affectation : pas de compatibilité pour l'affectation entre des types différents, cast obligatoires	Les opérations sont faites dans le type le plus fort L'affectation : cast implicites dans toutes les affectations entre types différents. Le type est celui de la variable qui reçoit.
Conversion de type :	**...L'opérateur de cast**
CBool(val) cast val en Boolean CByte CChar CDate CDbl CDec CInt CLng CObj CShort CSng CStr	NO (unsigned char) val (char) val NO (double)val NO (int)val (long)val (type*) ptr (short)val (float)val (char*)val

Opérateurs arithmétiques

affectation :	... identique mais
Dim n as Integer = 0 n = 100; En VB.NET c'est le résultat d'une opération qui donne le type de l'expression n=1 (n/100) est de type Single	int n=0; n=100 En C l'opération est faite dans le type de l'opérande le plus fort n=1 (n/100) est de type int

addition et soustraction :	...Identique
`Dim n as Integer = 0` `n = n+10;`	`int n=0;` `  n=n+10;`
Multiplication	... Identique mais...

`Dim ia As Integer = 9`
`Dim ib As Integer = 6`
`//`
 	
`Console.WriteLine("var : {0},{1},`
`{0}*{1}={2}", ia, ib, ia * ib)`
	`int ia=9,ib=6;`
	
	`//Différence printf`
	
	`printf("var : %d,%d,%d*%d=%d",`
	`  ia,ib,ia,ib,ia*ib);`

**Division : / et **	uniquement /

`/ : retourne une valeur décimale`
 	
`\ : retourne une valeur  entière`
 	
`Quelques soit les types des opérandes`
	`Valeur décimale ou entière selon le type des opérandes`

Modulo : mod	différence : %

`Dim A As integer = 10`
`Dim B As integer = 2`
`Dim C As integer`
 	
`  C = A mod B`
	`int A=10,B=2,C;`
	`  C = A % B;`

Les affectations combinées : **+=, -=, *=, /=, \=**	.. Identique

Branchements

Sauts conditionnels	Différence

`' cas 1 :`
`If test-vrai Then`
`  'les instructions`
`End If`
 	
`' cas 2 :`
`If test-vrai Then`
`  'les instructionsA`
`Else`
`  ' Les instructionsB`
`End If`
 	
`' cas 3 :`
`If test1-vrai Then`
`  'les instructions1`
	`// le cas 1`
	`if (test-vrai){`
	`  // instructions`
	`}`
	
	`// le cas 2`
	`if (test-vrai){`
	`  // instructionsA`
	`}`
	`else{`
	`  // instructionsB`
	`}`
	`// le cas 3`
	`if (test1-vrai){`
	`  // instructionsA`
	`}`

```
ElseIf test2-vrai Then
   ' Les instructions2

ElseIf testN-vrai Then
   ' Les instructionsN
Else
   ' instructions par défaut
End If
```

```
elseif(test2-vrai){
  // instructionsB
}
elseif(testN-vrai){
  // instructionsN
}
else{
  // instructions par défaut
}
```

Aiguillage : select Case

```
Dim i As integer
i = CInt(Rnd() * 15)

' possibilité 1 :
Select Case i
   Case 0 : ' vaut 0
   Case 5 : ' vaut 5
   Case else : 'autre que 0 ou 5
End Select

' possibilité 2 :
Select Case i
   Case 0,1,2,3,4, 5 :
      ' vaut entre 0 et 5
   Case 5 To 10 :
      ' vaut entre 5 et 10
   Case else : 'vaut plus que 10
End Select

' possibilité 3 :
Select Case i
   Case Is < 5 :
      ' vaut entre 0 et 4
   Case Is = 5
      ' vaut 5
   Case Is > 5 :
      ' vaut entre 6 et 15
   Case else : 'vaut plus que 5
End Select
```

Différence : switch ( )

```
int i;
   i=rand()%16;

// uniquement possibilité 1
switch( i ){
   case 0 : //vaut 0
      break;

   case 5 : //vaut 5
      break;

   default :// autre que 0 ou 5
      break;
}
```

Opérateurs de comparaison

Egalité : =	Différence : ==
Différent de : <>	Différence : !=
inférieur, inférieur ou égal : <, <=	...identique
Supérieur, supérieur ou égal : >, >=	... identique

Opérateurs de comparaison logique

<u>le non : Not</u>

```
Dim a As Integer=Rnd()*10

If Not (a Mod 2)
  Console.WriteLine("a pair")
Else
  Console.WriteLine("a impair)
```

différence : !

```
int a=rand()%11;

 if( ! (a%2) ){
   printf("a pair");
 else
   printf("a impair");
```

<u>Le ET : And</u>

```
Dim d1 As Integer
Dim d2 As Integer

  ' tirage dés 1
  d1 = 1 + CInt(Rnd()*5)

  ' tirage dés 2
  d2 = 1 + CInt(Rnd()*5)

  ' Si les deux propositions
  ' sont vraies on gagne :
    If d1 = 1 And d2 = 6 Then
        Console.WriteLine("Gagné")
    End If
```

différence : &&

```
int d1,d2;


   d1=1+rand()%6;
   d2=1+rand()%6;

   // Si les deux propositions
   //sont vraies on gagne :
    if (d1 == 1 && d2 == 6 )
         printf("Gagné");
```

<u>Le OU : Or</u>

```
  ' si une seule condition est
  ' vraie on gagne :
  If d1 = 1 Or d2 = 6 Then
    Console.WriteLine("Gagné")
  End If
```

différence : | |

```
   // si une seule condition est
   // vraie on gagne :
   if (d1 == 1 | | d2 == 6 )
      printf("Gagné");
```

Boucles

<u>Boucle For</u>

```
Dim i As Integer

' Base :
For i=0 To 10
   Console.WriteLine("{0}",i)
Next i

' pas d'avancement
For i=0 To 100 Step 10
   Console.WriteLine("{0}",i)
Next i

' pour forcer la sortie
For i=0 To 100 Step 10
   Console.WriteLine("{0}",i)
   if (Rnd()*10>=5)
      Exit For
Next i
```

Boucle Tant que

```
' figure 1
While condition
   ' instructions
Wend

' figure 2
Do While condition
  ' instructions
Loop

' figure 3
Do
   ' instructions
Loop While condition

' figure 4 : jusqu'à vrai
Do Until condition
   ' instructions
Loop

' figure 5 : jusqu'à vrai
Do
   'instructions
Loop Until condition

' figure 6 :
Do
   'instructions
   If (Rnd()*100>50)
      Exit Do
Loop
```

Tableaux

Avoir un tableau taille fixe

```
' une dimension
Dim tab(5) As Integer
Dim i As Integer

For i=0 To 4
   tab(i)=Rnd()*256
Next

' possibilité de spécifier les
' bornes inférieure et supérieure
Dim tab (0 To 4) As Short
For i=0 To 4
   tab(i)=Rnd()*256
Next

' plusieurs dimensions
Dim mat(5,10) as Short
Dim mat(0 To 4, 0 To 9) As Short
```

Redimensionner un tableau (tableau dynamique)	

Chaînes de caractères

Récupérer une chaine et afficher chacune de ses lettres	

```
Dim s As String
Dim i As Integer

   s = Console.ReadLine()
   Console.WriteLine("chaine entrée : " & s)

   For i = 0 To s.Length - 1
     Console.WriteLine(s(i))
   Next
```

Structures

Avoir une structure	

```
' La définition d'une structure se
' fait obligatoirement en globale.
' Définir une structure :
Structure point
   Dim x As Integer
   Dim y As Integer
End Structure

' déclarer ensuite une structure
Dim p As point

' accéder aux éléments d'une
' structure
  p.x = Rnd()*800
  p.y = Rnd()*600
```

Fonctions

Subroutines (procédures : pas de retour)	

```
' modèle du prototype
[Public/Private]Sub NomS ([Paramètre])
   ' instructions
End Sub

Private Sub hello()
   Console.WriteLine("Hello")
End Sub
```

Fonctions, une valeur de retour

```
' modèle du prototype
[Public/Private] Function nomF([paramètres]) As TypeRetour
   'instructions
End Fonction

' Par défaut les fonctions sont publiques c'est à dire
globales dans le programme

Fonction rand(range As Integer) As Integer
   return CInt(Rnd()*range)
End Fonction
```

Fonctions d'entrées - sorties en mode console

Écrire sur la fenêtre console

```
Console.WriteLine("hello VB.NET")

' Traduction automatique, pas de
' format :

Dim val As Single = 5.5
   Console.WriteLine(val)

' si texte à ajouter :
Console.WriteLine("val={0}",val)

' possibilité de concaténer sans conversion avec
'l'opérateur de concaténation & :
 Console.WriteLine( val & " test " & val*2)
' affiche : 5,5 test 11
```

Récupérer une entrée utilisateur en mode console

```
Dim i As Integer
  i = Console.ReadLine()
  Console.WriteLine("entré {0}", i)

Dim f As Single=0,5
  Console.WriteLine(f)

  ' Attention, dans le fenêtre console entrer le
  ' nombre avec une virgule et non un point
  f += Console.ReadLine()
  Console.WriteLine("res : {0}", f)

Dim s As String
  s = Console.readLine()
  Console.WriteLine ("chaine entrée : " & s)
```

Autres fonctions utiles