

WINDEV

Classe : 2ÉME TSIG SO
Enseignante : Souha HADDAD

haddad.souha.ing@gmail.com



Partie 01 - A la découverte de WINDEV

Partie 02 - Les bases du WLangage

Partie 03 - Ma première base de données

Partie 04 - Application complète avec données

Partie 05 - Administrer une base HFSQL Client/Serveur

Partie 06 - Optimiser un projet

Partie 07 - Quelques fonctionnalités WLangage

Partie 08 - Annexes & Conclusion



Partie 02

Les bases de WLangage



Partie 02 - Présentation (1/1)

- Il est un modèle de puissance, de facilité et d'intuitivité.
- C'est un langage de programmation de 5^{ème} génération (L5G).
- Inclus dans les outils de développement WINDEV, WEBDEV et WINDEV Mobile.
- Il est propriétaire et ne peut être manipulé qu'avec les outils PC SOFT.
- Il est né en 1992 avec la première version de WINDEV.

Partie 02 - Caractéristiques (1/3)

Le WLangage est constitué de mots-clés qui sont:

- des fonctions
- des constantes prédéfinies
- des instructions de déclaration des variables
- des instructions de déclaration de fonction et procédure
- des mots-clés
- des variables d'état ...

Partie 02 - Caractéristiques (2/3)

- WLangage supporte le modèle objet .
- Tous les mot-clés du WLangage existent en anglais.
- Un débogueur est disponible pour le WLangage.

Exemple:

Pour afficher un message:

Info("Hello World")

Partie 02 - Variables

Dans un langage de programmation, une variable permet de conserver des données.

Ce sont des parties de la mémoire qui contiennent des chaînes, des nombres, etc.

Le WLangage propose deux types de variables :

- les types simples, déclarables directement
- les types avancés, qui incluent un ou plusieurs variables de type simple.

Les variables permettent d'effectuer des calculs mathématiques, de faire des comparaisons ou tout simplement de stocker des informations pour les retrouver plus tard.

Partie 02 - Variables

Dans un langage de programmation, une variable permet de conserver des données. Ce sont des parties de la mémoire qui contiennent des chaînes, des nombres, etc.

Une variable est représentée par :

- un nom : Nom donné à la variable afin de la manipuler par le langage.
- un type : Nature de la donnée stockée dans la variable .
- une valeur : Information stockée dans la variable.
- une portée : Limite d'utilisation de la variable dans le programme .

Déclaration d'une variable

Prix est un monétaire



Nom de
la variable



Type de
la variable

La portée est définie principalement par l'endroit où la variable est déclarée.

Les variables permettent d'effectuer des calculs mathématiques, de faire des comparaisons ou tout simplement de stocker des informations pour les retrouver plus tard.

Partie 02 - Variables

Initialisation d'une variable

```
Prix est un monétaire
```

```
moPrix est un monétaire
```

Déclarer une variable

Avant d'utiliser une variable, il faut tout d'abord la déclarer (c'est-à-dire la créer).

- Exemple d'une déclaration simple

Initialisation d'une variable

```
Prix = 500.32
```



Valeur de la variable

Partie 02 - Variables

Déclarer une variable(1/3)

Avant d'utiliser une variable, il faut tout d'abord la déclarer (c'est-à-dire la créer).

- Exemple d'une déclaration simple :

```
Prix est un monétaire
```

- Prix représente le nom de la variable.
- est un permet de déclarer la variable. Le WLangage utilise le langage naturel en français.
- monétaire correspond au type de la variable.

Exemple d'une déclaration multiple :

```
Nom, Prénom sont des chaînes
```

- Nom, Prénom représentent les noms des variables.
- sont des permet de déclarer un ensemble de variables.
- chaînes représente le type des variables.

Partie 02 - Variables

Déclarer une variable(2/3)

```
// Affectation d'une variable de type monétaire  
Prix = 1256.67  
// Affichage du contenu de la variable  
Trace(Prix)  
// Affectation d'une variable de type chaîne  
Nom = "Dupont"  
// Affichage du contenu de la variable  
Trace(Nom)
```

Dans ce code, les variables Prix et Nom sont affectées avec une valeur.


Pour lire et manipuler le contenu de la variable, il suffit d'utiliser le nom donné à la variable pour y accéder.

Dans cet exemple, le contenu des variables est affiché dans le volet "Trace du débogueur" grâce à la fonction **Trace**.

Partie 02 - Variables

Déclarer une variable(3/3)

Testons notre code :

1. Lancez le test du projet en cliquant sur  parmi les boutons d'accès rapide.
2. Le projet se lance en exécution et l'éditeur de code réapparaît.
3. Affichez si nécessaire le volet "Trace du débogueur" pour voir le résultat : sous le volet "Accueil", dans le groupe "Environnement", déroulez "Volets" et sélectionnez "Trace du débogueur".
4. Le volet "Trace du débogueur" contient le contenu de nos variables :

```
1256.67  
Dupont
```

Partie 02 - Variables

Les types de variables

Le type de la variable permet d'indiquer quelle est la forme de l'information que l'on stocke dans la variable.

Les types les plus basiques sont :

- booléen (Vrai ou Faux),
- chaîne ("Dupont"),
- entier (1234),
- monétaire (12.32),
- réel (7.766666),
- etc.



Important

Utilisez le type correspondant à l'information que vous voulez stocker. Vous optimisez ainsi l'utilisation de la mémoire mais surtout vous évitez des erreurs de calcul ou de traitement lors de l'utilisation des variables dans les fonctions du WLangage.

Partie 02 - Variables

La portée des variables

Il est possible de déclarer les variables à n'importe quel endroit dans le code. Cependant, en fonction de la position de sa déclaration, la variable n'est pas utilisable (ou visible) pour effectuer des traitements ou des calculs. On parle alors de portée des variables.

Il existe 2 types de portée :

- Globale.
- Locale.

Portée Globale

Le terme Global(e) signifie que la variable a une visibilité étendue dans le code. La variable est visible en dehors de l'endroit où elle a été déclarée. Il y a plusieurs niveaux de Globalité :

- niveau Projet et Collection de procédures,
- niveau Fenêtre, Fenêtre Mobile, Page, Etat,
- niveau Champ.



Note

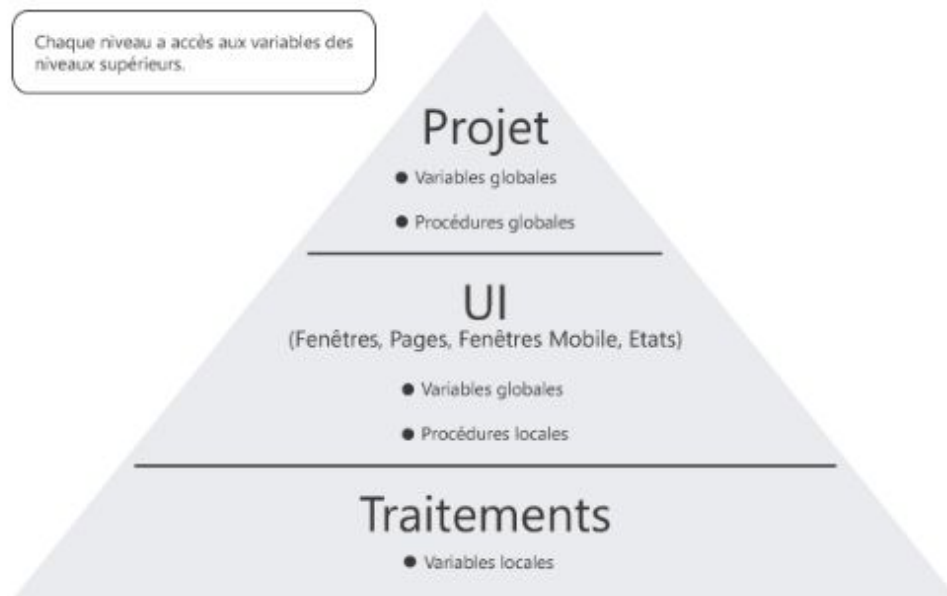
Par défaut, les variables globales sont colorées en bleu dans l'éditeur de code. Pour identifier les variables globales et leur portée, il est recommandé d'utiliser une norme d'écriture.

Partie 02 - Variables

Portée Locale

Le terme Local(e) signifie que la variable a une visibilité très limitée dans le code. La variable est visible uniquement dans le traitement où elle a été déclarée. Cela permet ainsi de restreindre l'utilisation de la variable au traitement ou à l'événement.

Schéma récapitulatif de la portée



Partie 02 - Variables

Opérations simples sur les variables

Pour effectuer des calculs sur des variables, le WLangage met à disposition les opérateurs mathématiques classiques :

- + pour faire une addition.
- - pour faire une soustraction.
- * pour faire une multiplication.
- / pour faire une division.

D'autres opérateurs peuvent être utilisés pour réaliser des calculs :

- ++ pour incrémenter de 1 (ajouter 1 à la variable).
- -- pour décrémenter de 1 (soustraire 1 à la variable).
- += pour affecter en additionnant une valeur.
- -= pour affecter en soustrayant une valeur.

Partie 02 - Variables

```
// Déclaration de variables
Compteur est un entier
Vi est un entier
Res est un numérique


// Affectation
Compteur = 10
Vi = 3

// Utilisation des opérateurs
Compteur = Compteur + 3 // Compteur vaut 13
Trace(Compteur)
Compteur ++ // Compteur vaut 14
Trace(Compteur)
Compteur -= 8 // Compteur vaut 6
Trace(Compteur)
Compteur = Compteur * Vi // Compteur vaut 18
Trace(Compteur)
Res = Compteur / 5 // Res vaut 3.6
Trace(Res)
```

Ce code permet de faire diverses opérations et affiche à chaque fois le résultat dans la fenêtre de trace.

Partie 02 - Variables

Testons notre code :

1. Lancez le test du projet en cliquant sur  parmi les boutons d'accès rapide.
2. Affichez si nécessaire le volet "Trace du débogueur" pour voir le résultat : sous le volet "Accueil", dans le groupe "Environnement", déroulez "Volets" et sélectionnez "Trace du débogueur".
3. Le volet "Trace du débogueur" contient la valeur de la variable Compteur pour chaque opération réalisée :



```
13
14
6
18
3.6
```

Des opérateurs de comparaison sont également disponibles :

- < inférieur strict
- > supérieur strict
- <= inférieur ou égal
- >= supérieur ou égal
- <> différent de
- = égal à

Partie 02 - Variables

Conseils :

- Il est très pratique de nommer les variables avec des noms longs (et éviter les noms très courts i, j, k par exemple). Lors de la relecture du programme, il est ainsi possible de se rappeler à quoi sert la variable.
- Pour définir le nom des variables, le WLangage accepte tous les caractères Unicode, y compris les accents. Autant en profiter pour plus de lisibilité ! Attention toutefois : certains caractères sont interdits : espace, =, point, virgule, etc.
- Il est très important de donner le bon type à la variable en fonction de son utilisation. Par exemple, pour stocker une suite de chiffres, il peut être nécessaire :
 - d'utiliser une variable de type numérique si cette variable doit être utilisée pour des calculs.
 - d'utiliser une variable de type chaîne si cette variable permet de stocker des chiffres sans effectuer de calculs (par exemple pour mémoriser le numéro de sécurité sociale).

Partie 02 - Variables

Détail d'un type de variable : les variables de type Chaîne

Les variables de type Chaîne sont un des types de variables les plus utilisés.

Nous allons détailler une partie des fonctionnalités disponibles sur ce type de variable.

Le type Chaîne

Nous savons déjà que le type Chaîne permet de stocker et manipuler les caractères et les chaînes de caractères.

Nous avons déjà vu comment initialiser une variable de type chaîne :

```
Nom est une chaîne  
//Affectation d'une variable de type chaîne  
Nom = "Dupont"
```



Note

En WLanguage, le caractère " (double quote) est le caractère utilisé pour délimiter une chaîne de caractères. Dans l'exemple ci-dessus, les doubles quotes sont utilisées pour affecter la valeur Dupont à la variable Nom.

Partie 02 - Variables

Il n'est pas nécessaire de déclarer la longueur de la chaîne : le WLangage adapte automatiquement cette longueur lors de l'utilisation de la variable.



Astuce

Pour initialiser une variable de type chaîne avec un texte sur plusieurs lignes, utilisez la syntaxe suivante :

```
<Nom variable> = "  
    <Texte de la ligne 1>  
    <Texte de la ligne 2>  
"
```

Par exemple :

```
MaChaîne est une chaîne  
MaChaîne = "  
    Exemple de  
    chaîne multi-ligne  
"
```

Il est également possible d'affecter une variable de type chaîne avec le contenu d'un champ manipulant des chaînes. Ainsi, le code permettant d'affecter une variable de type chaîne avec le contenu d'un champ de saisie est le suivant :

Partie 02 - Variables

Il est également possible d'affecter une variable de type chaîne avec le contenu d'un champ manipulant des chaînes. Ainsi, le code permettant d'affecter une variable de type chaîne avec le contenu d'un champ de saisie est le suivant :

```
Nom est une chaîne  
//Affectation d'une variable de type chaîne  
//avec le contenu du champ de saisie SAI_Nom  
Nom = SAI_Nom
```

Construction d'une chaîne

Comme nous l'avons vu précédemment, la déclaration et l'affectation d'une chaîne est très simple. Par exemple :

```
MaChaineConstruite est une chaîne  
NomProduit est une chaîne = "WINDEV"
```

MaChaineConstruite est une chaîne

NomProduit est une chaîne = 'WINDEV'

// Concaténation

MaChaineConstruite = "Mon outil de développement, c'est " + NomProduit + "!"

Trace("Concaténation : " + MaChaineConstruite)

// ChaîneConstruit

MaChaineConstruite = ChaîneConstruit("Mon outil de développement, c'est %s!", NomProduit)

Trace("ChaîneConstruit : " + MaChaineConstruite)


// Saisie directe de la variable (syntaxe [%s %s])

MaChaineConstruite = "Mon outil de développement, c'est [%sNomProduit%s]!"

Trace("Saisie directe : " + MaChaineConstruite)

Partie 02 - Variables

Testons notre code :

1. Lancez le test du projet en cliquant sur  parmi les boutons d'accès rapide. Le projet se lance et l'éditeur de code réapparaît.
2. Affichez si nécessaire le volet "Trace du débogueur" pour voir le résultat : sous le volet "Accueil", dans le groupe "Environnement", déroulez "Volets" et sélectionnez "Trace du débogueur".
3. Le volet "Trace du débogueur" contient nos exemples de concaténation :

```
Concaténation : Mon outil de développement, c'est WINDEV !  
ChaîneConstruit : Mon outil de développement, c'est WINDEV !  
Saisie directe : Mon outil de développement, c'est WINDEV !
```


Partie 02 - LES INSTRUCTIONS CONDITIONNELLES

L'instruction SI

Cette instruction permet d'exécuter un traitement ou un autre en fonction du résultat d'une expression. Si l'expression est vérifiée, un traitement est lancé ; si l'expression n'est pas vérifiée, un autre traitement peut être lancé.

L'instruction SI s'utilise comme ci-dessous :

```
SI <Expression à vérifier> ALORS  
    Traitement à exécuter dans le cas où l'expression est vérifiée  
SINON  
    Traitement à exécuter dans le cas contraire  
FIN
```

Remarque : L'instruction SI peut également être présentée sous forme d'une seule ligne grâce à l'instruction "? ... SINON" :

```
<Expression à vérifier> ? <Traitement si expression vérifiée>  
    SINON <Traitement dans le cas contraire>
```

NombreAléatoire est un entier

// Initialise le générateur de nombre aléatoire

InitHasard()

// Prend un nombre au hasard entre 100 et 4000

NombreAléatoire = Hasard(100, 4000)

Trace("Valeur du nombre aléatoire " + NombreAléatoire)

// Vérifie si ce nombre est supérieur strictement à 2000

SI NombreAléatoire > 2000 ALORS

Trace("Nombre supérieur à 2000")

SINON

Trace("Nombre inférieur ou égal à 2000")

FIN

// Instruction conditionnelle monoligne

Trace(NombreAléatoire > 2000 ? "Nombre supérieur à 2000" SINON "Nombre inférieur ou égal à 2000")

Partie 02 - LES INSTRUCTIONS CONDITIONNELLES

```
// Condition composée (mots-clés ET et OU)  
Condition1 est un booléen = Faux  
Condition2 est un booléen = Vrai  
// - mot-clé ET  
SI Condition1 ET Condition2 ALORS  
    Trace("Condition1 ET Condition2 : <vrai>")  
SINON  
    Trace("Condition1 ET Condition2 : <faux>")  
FIN
```

```
Condition1 ET Condition2 : <faux>
```

Partie 02 - LES INSTRUCTIONS CONDITIONNELLES

Condition composée avec le mot-clé OU : Permet de tester si l'une ou l'autre des conditions sont respectées. Là aussi, les

```
Condition1 est un booléen = Faux
```

```
Condition2 est un booléen = Vrai
```

```
SI Condition1 OU Condition2 ALORS
```

```
    Trace('Condition1 OU Condition2 : <vrai>')
```

```
SINON
```

```
    Trace('Condition1 OU Condition2 : <faux>')
```

```
FIN
```

```
Condition1 OU Condition2 : <vrai>
```

Partie 02 - LES INSTRUCTIONS CONDITIONNELLES

L'instruction SELON

Cette instruction permet d'évaluer une expression et d'exécuter un traitement pour chaque valeur possible de l'expression.

L'instruction SELON s'utilise comme ci-dessous :

```
SELON <Expression>  
CAS Valeur 1 :  
    Traitement 1  
CAS Valeur 2 :  
    Traitement 2  
...  
CAS Valeur N :  
    Traitement N  
  
AUTRES CAS  
    Traitement  
  
FIN
```

Partie 02 - LES INSTRUCTIONS CONDITIONNELLES

Nous allons écrire un petit code WLangage afin de tester cette instruction :

```
// Par défaut, la variable de type Date est initialisée avec la date du jour
LaDateDujour est une Date

// Teste le jour de la date
SELON LaDateDujour Jour
    // 1er jour du mois
    CAS 1: Trace('Nous sommes le premier jour du mois.')
    // 15ème jour du mois
    CAS 15: Trace('Nous sommes le 15 du mois.')
    // Pour toutes les autres jours, affiche la date
    AUTRES CAS: Trace('Nous sommes le ' + LaDateDujour.VersChaîne())
FIN
```

```
Nous sommes le : 28/11/2021
```

Partie 02 - LES INSTRUCTIONS CONDITIONNELLES

Nous allons écrire un petit code WLangage afin de tester cette instruction :



Note

Le WLangage est composé de fonctions et de propriétés. Les fonctions peuvent attendre des paramètres et renvoient des résultats. Les propriétés sont directement appliquées aux champs ou aux variables grâce à la syntaxe :

```
<Nom du champ ou de la variable>.<Nom de la propriété>
```

Dans notre exemple, la propriété **Jour** est utilisée sur la variable **LaDateDuJour** pour obtenir le jour de la date.

Partie 02 - LES INSTRUCTIONS CONDITIONNELLES

```
SELON LaDateDuJour.Mois
```

```
// Les mois 1, 2 et 3
```

```
CAS 1, 2, 3 : Trace('Nous sommes au premier trimestre de l'année.')
```

```
// Les mois compris entre 4 et 6
```

```
CAS 4 <= * <= 6 : Trace('Nous sommes au deuxième trimestre de l'année.')
```

```
// Les mois après le sixième mois
```

```
CAS > 6 : Trace('Nous sommes au second semestre de l'année.')
```

```
FIN
```


Partie 02 - LES INSTRUCTIONS ITÉRATIVES

Les instructions de boucle permettent d'exécuter un traitement de manière récurrente. Selon le nombre d'occurrences (connu ou pas), on utilise une instruction de boucle particulière. Il existe plusieurs instructions pour faire des boucles :

- POUR ...
- BOUCLE ...
- TANTQUE ...

L'instruction POUR

L'instruction POUR est utilisée lorsque l'on connaît le nombre d'occurrences à traiter. Cette instruction permet de gérer le nombre d'occurrences à l'aide d'une variable dans laquelle on va compter les passages effectués dans la boucle.

La syntaxe de l'instruction POUR est la suivante :

```
POUR Indice = Valeur de départ A Valeur de fin  
    Traitement à exécuter  
FIN
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction POUR (1/10)

Pour tester cette instruction, nous allons saisir un code WLangage permettant de remplir un tableau d'entiers avec les nombres pairs compris entre 1 et 10. Nous listerons ensuite le contenu du tableau. Nous allons écrire ce code en plusieurs étapes.

1. Supprimez si nécessaire le code WLangage présent dans l'événement "Initialisation" du projet.
2. Nous allons tout d'abord remplir un tableau d'entiers de 1 à 10. Copiez le code suivant :

```
TableauNombres est un tableau d'entiers

POUR Indice = 1 À 10
    // Affiche l'indice
    Trace("Tour de boucle | Valeur de l'indice : [%Indice%]")
    TableauNombres.Ajoute(Indice)
    Trace("Ajout du nombre [%Indice%] dans le tableau")
FIN
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction POUR (2/10)

3. Testons immédiatement ce code : cliquez sur GO parmi les boutons d'accès rapide.
4. Affichez si nécessaire le volet "Trace du débogueur" pour voir le résultat : sous le volet "Accueil", dans le groupe "Environnement", déroulez "Volets" et sélectionnez "Volets" puis "Trace du débogueur".
5. Le volet "Trace du débogueur" contiendra les messages du tour de boucle et de l'ajout dans le tableau pour les 10 indices.

```
Tour de boucle | Valeur de l'indice : 1  
Ajout du nombre 1 dans le tableau  
Tour de boucle | Valeur de l'indice : 2  
Ajout du nombre 2 dans le tableau  
Tour de boucle | Valeur de l'indice : 3  
Ajout du nombre 3 dans le tableau  
Tour de boucle | Valeur de l'indice : 4  
Ajout du nombre 4 dans le tableau  
Tour de boucle | Valeur de l'indice : 5  
Ajout du nombre 5 dans le tableau  
Tour de boucle | Valeur de l'indice : 6  
Ajout du nombre 6 dans le tableau  
Tour de boucle | Valeur de l'indice : 7  
Ajout du nombre 7 dans le tableau  
Tour de boucle | Valeur de l'indice : 8  
Ajout du nombre 8 dans le tableau  
Tour de boucle | Valeur de l'indice : 9  
Ajout du nombre 9 dans le tableau  
Tour de boucle | Valeur de l'indice : 10  
Ajout du nombre 10 dans le tableau
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction POUR (3/10)

Nous allons modifier ce code pour sortir de la boucle si l'indice vaut 5. Dans ce cas, nous allons tester la valeur de l'indice et si celle-ci vaut 5, il suffira d'utiliser le mot-clé SORTIR pour arrêter la boucle.

1. Le code devient :

```
TableauNombres est un tableau d'entiers

POUR Indice = 1 À 10
    //Affiche l'indice
    Trace("Tour de boucle | Valeur de l'indice : [%Indice%]")

    SI Indice = 5 ALORS
        Trace("Condition de sortie atteinte")
        SORTIR
    FIN
    TableauNombres.Ajoute(Indice)
    Trace("Ajout du nombre [%Indice%] dans le tableau")
FIN
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction POUR (4/10)

2. Testons immédiatement ce code : cliquez sur GO parmi les boutons d'accès rapide.
3. Le volet "Trace du débogueur" contiendra les messages du tour de boucle et de l'ajout dans le tableau pour les 5 premiers indices.

```
Tour de boucle | Valeur de l'indice : 1  
Ajout du nombre 1 dans le tableau  
Tour de boucle | Valeur de l'indice : 2  
Ajout du nombre 2 dans le tableau  
Tour de boucle | Valeur de l'indice : 3  
Ajout du nombre 3 dans le tableau  
Tour de boucle | Valeur de l'indice : 4  
Ajout du nombre 4 dans le tableau  
Tour de boucle | Valeur de l'indice : 5  
Condition de sortie atteinte
```

TableauNombres est un tableau d'entiers

POUR Indice = 1 À 10

Trace("Tour de boucle | Valeur de l'indice : [%Indice%]")

SI Indice = 5 ALORS

Trace("Condition de sortie atteinte")

SORTIR

FIN

SI EstImpair(Indice) = Vrai ALORS

Trace("L'indice est impair : passe à l'itération suivante")

CONTINUER

FIN

TableauNombres.Ajoute(Indice)

Trace("Ajout du nombre [%Indice%] dans le tableau")

FIN

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction POUR (6/10)

2. Testons immédiatement ce code : cliquez sur GO parmi les boutons d'accès rapide.
3. Le volet "Trace du débogueur" contiendra le message du tour de boucle pour les 5 premiers indices. Pour les nombres impairs, un message indique que le code d'ajout n'est pas exécuté. Le message d'ajout dans le tableau sera affiché uniquement pour les nombres pairs.

```
Tour de boucle | Valeur de l'indice : 1  
L'indice est impair : passe à l'itération suivante  
Tour de boucle | Valeur de l'indice : 2  
Ajout du nombre 2 dans le tableau  
Tour de boucle | Valeur de l'indice : 3  
L'indice est impair : passe à l'itération suivante  
Tour de boucle | Valeur de l'indice : 4  
Ajout du nombre 4 dans le tableau  
Tour de boucle | Valeur de l'indice : 5  
Condition de sortie atteinte
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction POUR (7/10)

Il nous reste à ajouter le code permettant de parcourir le tableau afin de lister les indices enregistrés. Nous allons encore utiliser une boucle de type POUR. Les extrémités du parcours seront :

- le premier élément du tableau. Cet élément correspond à 1
 - le dernier élément du tableau. Comme nous ne le connaissons pas, nous allons utiliser la propriété Occurrence. Cette propriété, utilisée sur un tableau, permet de connaître le nombre d'éléments du tableau (et donc l'indice du dernier élément).
1. Ajoutez le code suivant :

```
POUR Indice = 1 À TableauNombres.Occurrence
    //Affiche l'indice et la valeur correspondante dans le tableau
    Trace("Indice : [%Indice%] | Valeur : " + TableauNombres[Indice])
FIN
```


Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction POUR (8/10)

2. Ce code peut être optimisé. En effet, avec cette syntaxe, le nombre d'éléments du tableau est réévalué à chaque tour de boucle. Si le nombre d'éléments du tableau évolue (des éléments sont supprimés par exemple), la valeur finale de la variable de contrôle s'adaptera. Dans notre exemple, le tableau n'est pas modifié pendant le parcours : il est possible d'utiliser le mot-clé `_A_` qui permet de ne pas recalculer le nombre d'occurrences à chaque tour. Le code devient :

```
POUR Indice = 1 À TableauNombres.Occurrence
  //Affiche l'indice et la valeur correspondante dans le tableau
  Trace("Indice : [%Indice%] | Valeur : " + TableauNombres[Indice])
FIN
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction POUR (9/10)

3. Testons immédiatement ce code : cliquez sur GO parmi les boutons d'accès rapide.
4. Le volet "Trace du débogueur" contiendra le message affiché pour les 5 premiers indices. Pour les nombres impairs, un message indique que le code d'ajout n'est pas exécuté. Le contenu du tableau est ensuite listé.

```
Tour de boucle | Valeur de l'indice : 1
L'indice est impair : passe à l'itération suivante
Tour de boucle | Valeur de l'indice : 2
Ajout du nombre 2 dans le tableau
Tour de boucle | Valeur de l'indice : 3
L'indice est impair : passe à l'itération suivante
Tour de boucle | Valeur de l'indice : 4
Ajout du nombre 4 dans le tableau
Tour de boucle | Valeur de l'indice : 5
Condition de sortie atteinte
Indice : 1 | Valeur : 2
Indice : 2 | Valeur : 4
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction POUR (10/10)

Remarque : Il est possible de définir un pas d'incrémentation de l'indice grâce au mot-clé PAS. Par exemple, le code suivant exécute 2000 fois le traitement et la variable Indice diminue de 10 en 10 :

```
// Boucle POUR allant de 2000 à 1, en allant de 10 en 10  
POUR Indice = 2000 À 1 PAS -10  
    // Affiche l'indice  
    Trace(Indice)  
FIN
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction BOUCLE

L'instruction BOUCLE est utilisée pour faire des boucles lorsque le nombre d'occurrences à traiter n'est pas connu. Dans ce cas, il faut utiliser un test sur une condition pour sortir de la boucle.

La syntaxe de l'instruction BOUCLE est la suivante :

```
BOUCLE  
    Traitement à exécuter  
    SI <Condition à vérifier> ALORS SORTIR  
FIN
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction BOUCLE (1/4)

Nous allons écrire un petit code WLangage afin de tester cette instruction :

1. Supprimez si nécessaire le code WLangage présent dans l'événement "Initialisation" du projet et copiez le code suivant.

Ce code soustrait 1 à un entier jusqu'à ce que la valeur de l'entier soit nulle :

```
Compteur est un entier
Compteur = 10
BOUCLE
    Trace("Tour de boucle | Valeur : [%Compteur%]")
    Compteur--
    SI Compteur = 0 ALORS SORTIR
FIN
Trace("Fin de boucle")
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction BOUCLE (2/4)

2. Testons immédiatement ce code : cliquez sur GO parmi les boutons d'accès rapide.
3. Le volet "Trace du débogueur" contient le message affiché pour les 10 tours de boucle.

```
Tour de boucle | Valeur : 10  
Tour de boucle | Valeur : 9  
Tour de boucle | Valeur : 8  
Tour de boucle | Valeur : 7  
Tour de boucle | Valeur : 6  
Tour de boucle | Valeur : 5  
Tour de boucle | Valeur : 4  
Tour de boucle | Valeur : 3  
Tour de boucle | Valeur : 2  
Tour de boucle | Valeur : 1  
Fin de boucle
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction BOUCLE (3/4)

Une autre possibilité de test dans les boucles est l'utilisation du mot-clé TANTQUE. Dans ce cas, la syntaxe de l'instruction BOUCLE est la suivante :

```
Compteur est un entier
Compteur = 10
BOUCLE
    Trace("Tour de boucle | Valeur : [%Compteur%]")
    Compteur = Compteur - 1
À FAIRE TANTQUE Compteur > 0
Trace("Fin de boucle")
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction BOUCLE (4/4)

L'instruction BOUCLE permet d'avoir le même comportement qu'une instruction POUR : il suffit d'utiliser la syntaxe avec sortie selon le nombre d'itérations :

```
BOUCLE (<Nombre d'itérations>)  
...  
FIN
```



Astuce

Dans notre exemple, le code peut ainsi devenir :

```
Compteur est un entier  
Compteur = 10  
BOUCLE(10)  
    Trace("Tour de boucle | Valeur : [%Compteur%]")  
    Compteur --  
FIN  
Trace("Fin de boucle")
```


Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction TANTQUE

L'instruction TANTQUE fonctionne sur le même principe que l'instruction BOUCLE. La différence réside dans le fait que le test de la condition de sortie est effectué AVANT l'exécution du code de la boucle. Ce test permet de comparer une variable à une valeur donnée. Cette variable commence à une valeur de départ et est modifiée dans la boucle jusqu'à arriver à la valeur qui provoque la sortie de la boucle.

La syntaxe de l'instruction TANTQUE est la suivante :

```
<Initialisation de la variable à sa valeur de début>  
TANTQUE <Comparaison de la variable à sa valeur de fin>  
    Traitement à exécuter  
    <Modification de la variable>  
FIN
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction TANTQUE (1/2)

Nous allons écrire un petit code WLangage afin de tester cette instruction :

1. Supprimez si nécessaire le code WLangage présent dans l'événement "Initialisation" du projet et copiez le code suivant.

Ce code additionne 1 à un entier tant que l'entier est inférieur à 10.

```
Compteur est un entier = 1
TANTQUE Compteur <= 10
    Trace("Tour de boucle | Valeur : [%Compteur%]")
    // Traitement à exécuter
    Compteur ++
FIN
Trace("Fin de boucle")
```

Partie 02 - LES INSTRUCTIONS ITÉRATIVES

L'instruction TANTQUE (2/2)

2. Testons immédiatement ce code : cliquez sur GO parmi les boutons d'accès rapide.
3. Le volet "Trace du débogueur" contient le message affiché pour les 10 tours de boucle.

```
Tour de boucle | Valeur : 1  
Tour de boucle | Valeur : 2  
Tour de boucle | Valeur : 3  
Tour de boucle | Valeur : 4  
Tour de boucle | Valeur : 5  
Tour de boucle | Valeur : 6  
Tour de boucle | Valeur : 7  
Tour de boucle | Valeur : 8  
Tour de boucle | Valeur : 9  
Tour de boucle | Valeur : 10  
Fin de boucle
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Tableau

Les tableaux sont un type de variable très utilisé.

Un tableau est un type structuré qui permet de regrouper une série d'éléments de même type. Chaque élément du tableau est accessible directement par son numéro d'ordre (indice).

Des fonctions WLangage spécifiques permettent de manipuler les tableaux et leurs éléments.

Déclaration

La déclaration d'une variable de type tableau se fait de la façon suivante :

```
<Nom du tableau> est un tableau de <Type des éléments du tableau>
```

Par exemple :

```
MonTabChaîne est un tableau de chaînes  
MonTabEntier est un tableau d'entiers
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Tableau

Remplissage d'un tableau et accès aux éléments

Lors de sa déclaration, le tableau est vide. L'ajout d'éléments est réalisé par la fonction <Tableau>.Ajoute grâce à la syntaxe suivante :

```
<Nom du tableau>.Ajoute(<Valeur de l'élément>)
```

Pour accéder aux éléments d'un tableau, il suffit d'utiliser la syntaxe suivante :

```
<Nom du tableau> [<Indice de l'élément>]
```



Important

Les indices des éléments du tableau commencent à 1.

Partie 02 - TOUR D'HORIZON DES VARIABLES

Tableau

Pour tester l'ajout d'éléments dans un tableau, nous allons tester un petit exemple de code :

1. Supprimez le code saisi dans l'événement "Initialisation" du projet (Ctrl + A pour tout sélectionner, et Suppr pour supprimer).
2. Saisissez le code suivant :

```
//Création d'un tableau de chaînes  
MonTableau est un tableau de chaînes|  
  
//Ajout d'éléments  
MonTableau.Ajoute("WINDEV")  
MonTableau.Ajoute("WEBDEV")  
MonTableau.Ajoute("WINDEV Mobile")  
  
//Affichage du contenu du troisième élément  
Trace("Valeur de l'élément 3 : [%MonTableau[3] %]")
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Tableau

3. Testons immédiatement ce code : cliquez sur GO parmi les boutons d'accès rapide.
4. Affichez si nécessaire le volet "Trace du débogueur" pour voir le résultat : sous le volet "Accueil", dans le groupe "Environnement", déroulez "Volets" et sélectionnez "Trace du débogueur".
5. Le volet "Trace du débogueur" contient les informations suivantes :

Valeur de l'élément 3 : WINDEV Mobile

Initialisation rapide d'un tableau

Pour initialiser un tableau, il est également possible d'utiliser la syntaxe suivante :



Astuce

```
//Déclare un tableau  
TableauJour est un tableau de chaînes  
//Initialisation avec  
//les noms des jours de la semaine  
TableauJour = ["Lundi", "Mardi", "Mercredi", "Jeudi",  
"Vendredi", "Samedi", "Dimanche"]
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Parcours d'un Tableau

Pour parcourir les éléments d'un tableau, le WLangage propose deux méthodes :

- Une boucle classique de type POUR .. A.
- Une boucle de type POUR TOUT.
- Dans le cas d'une boucle classique de type POUR .. A .., il est nécessaire de donner la valeur de l'indice de départ (1) et de l'indice d'arrivée.

1. Par exemple, ajoutez le code suivant dans l'événement "Initialisation" du projet :

```
POUR Indice = 1 À MonTableau.Occurrence  
    Trace("Valeur de l'élément [%Indice%]: [%MonTableau[Indice]%]")  
FIN
```


Partie 02 - TOUR D'HORIZON DES VARIABLES

Parcours d'un Tableau

1. Dans ce code, l'indice de départ est 1, l'indice d'arrivée est donné par la propriété **Occurrence** du tableau.
2. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
3. Le volet "Trace du débogueur" contient les informations suivantes :

```
Valeur de l'élément 1 : WINDEV  
Valeur de l'élément 2 : WEBDEV  
Valeur de l'élément 3 : WINDEV Mobile
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Parcours d'un Tableau

Le WLangage propose un type de boucle beaucoup plus simple et tout aussi performant : la boucle POUR TOUT. Nous l'avons déjà vue sur les chaînes. Ce type de boucle s'applique également sur les tableaux. Dans ce cas, pas besoin d'indiquer l'indice de départ ou d'arrivée.

1. Par exemple, remplacez le code de la boucle POUR par le code suivant dans l'événement "Initialisation" du projet :

```
POUR TOUT ÉLÉMENT UneChaîne, Indice DE MonTableau  
    Trace("Valeur de l'élément [%Indice%]: [%UneChaîne%]")  
FIN
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Parcours d'un Tableau

Dans ce cas, il suffit d'indiquer le nom de la variable qui correspond à l'élément du tableau et celle qui correspond à l'indice.

1. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
2. Le volet "Trace du débogueur" contient les informations suivantes :

```
Valeur de l'élément 1 : WINDEV  
Valeur de l'élément 2 : WEBDEV  
Valeur de l'élément 3 : WINDEV Mobile
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Parcours d'un Tableau

Dans ce cas, il suffit d'indiquer le nom de la variable qui correspond à l'élément du tableau et celle qui correspond à l'indice.

1. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
2. Le volet "Trace du débogueur" contient les informations suivantes :

```
Valeur de l'élément 1 : WINDEV  
Valeur de l'élément 2 : WEBDEV  
Valeur de l'élément 3 : WINDEV Mobile
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Tableau associatif

Un tableau associatif est un type "avancé" de tableau : il permet de regrouper une série d'éléments du même type. Chaque élément du tableau est indexé sur n'importe quel type d'information (et non plus simplement sur un indice numérique, comme dans les autres types de tableau).

Les Tableaux associatifs (avec ou sans doublons) sont manipulables en WLangage.

- Vous voulez tester ? Rien de plus simple.
 1. Supprimez le code saisi dans l'événement "Initialisation" du projet (Ctrl + A pour tout sélectionner, et Suppr pour supprimer).
 2. Saisissez le code suivant :

Partie 02 - TOUR D'HORIZON DES VARIABLES

Tableau associatif

```
//Création d'un tableau associatif de dates
MonTableau est un tableau associatif de Dates

//Ajout d'éléments
MonTableau["Marc"] = "19820201"
MonTableau["Anne"] = "19840604"

//Affichage du contenu de l'élément ayant pour clé "Marc"
Trace("Valeur de l'élément ayant pour clé "Marc": " + MonTableau["Marc"])

//Parcours du tableau
POUR TOUT ÉLÉMENT Valeur, Clé, Indice DE MonTableau
    Trace("Valeur de l'élément [%Indice%] qui a pour clé [%Clé%]: [%Valeur%]")
FIN
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Tableau associatif

3. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
4. Le volet "Trace du débogueur" contient les informations suivantes :

```
Valeur de l'élément ayant pour clé "Marc" : 19820201  
Valeur de l'élément 1 qui a pour clé Marc : 19820201  
Valeur de l'élément 2 qui a pour clé Anne : 19840604
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Variables de type Date, Heure et Durée

En dehors des types simples (booléen, entier, réel, ...), le WLangage propose plusieurs types de variables qui permettent de prendre en compte beaucoup de spécificités. Nous allons passer rapidement en revue ces différents types de variables avec pour chacun un petit exemple pratique. Nous vous conseillons de consulter l'aide en ligne du type proposé pour plus de détails.

Partie 02 - TOUR D'HORIZON DES VARIABLES

Date

Le WLangage propose différents types de variables permettant de gérer les dates, les heures et les durées.

Pour gérer les dates, le WLangage propose le type Date. De nombreuses fonctions et propriétés permettent de manipuler les dates.

Pour tester la manipulation des dates, nous allons tester un petit exemple de code :

1. Supprimez le code saisi dans l'événement "Initialisation" du projet (Ctrl + A pour tout sélectionner, et Suppr pour supprimer).
2. Saisissez le code suivant :

```
MaDate est une Date
```

Cette ligne de code permet de déclarer une variable de type Date. Par défaut, la variable est initialisée avec la date du jour.

3. Maintenant, voyons comment affecter une date.

```
MaDate = Hier()  
MaDate = "20210929"
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Date

- via une fonction WLangage. Dans cet exemple, la fonction Hier est utilisée. Le WLangage propose de nombreuses fonctions permettant de connaître le premier ou dernier jour du mois, de l'année, ...
 - via une chaîne de caractères contenant la date au format "AAAAMMJJ".
4. Les paramètres de la date peuvent également être modifiés à l'aide de propriétés WLangage. Saisissez le code suivant :

```
// Modification des paramètres de la date  
MaDate.Année = 1985  
MaDate.Mois = 10  
MaDate.Jour = 26
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Date

La fonction <Date>.VersChaîne permet de choisir le format d'affichage d'une date. Par exemple, saisissez le code suivant :

```
Trace(MaDate.VersChaîne())  
Trace(MaDate.VersChaîne("jjjj JJ Mmmm AAAA"))  
Trace(MaDate.VersJourEnLettre())
```

5. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
6. Le volet "Trace du débogueur" contient les informations suivantes :

```
26/10/1985  
Samedi 26 Octobre 1985  
samedi
```

- l'affichage en clair de la date.
- l'affichage en clair de la date selon un masque.
- la récupération du jour en lettres correspondant à la date.

Partie 02 - TOUR D'HORIZON DES VARIABLES

Date

8. Il est également possible d'utiliser des opérateurs de comparaison entre des dates. Saisissez le code suivant :

```
SI MaDate < DateDujour() ALORS  
    Trace("La date mémorisée est antérieure à la date du jour")  
FIN
```

9. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
10. Le volet "Trace du débogueur" contient les informations suivantes :

```
26/10/1985  
Samedi 26 Octobre 1985  
samedi  
La date mémorisée est antérieure à la date du jour
```

```
// Déclaration d'une heure
MonHeure est une Heure // Par défaut, l'heure est initialisée avec l'heure actuelle

// Affectation d'une heure
// - via une fonction WLangage
MonHeure = Maintenant()
// - fixée dans le code (par exemple 15h25)
MonHeure = "1525"

// Modification des paramètres de l'heure
MonHeure.Heure = 12
MonHeure.Minute = 0
MonHeure.Seconde = 0

// Affichage en clair de l'heure
Trace(MonHeure.VersChaîne())
// Affichage en clair de l'heure selon un masque
Trace(MonHeure.VersChaîne("HH:mm:ss"))

// Comparaison de 2 heures
SI MonHeure < Maintenant() ALORS
    Trace("L'heure mémorisée est antérieure à l'heure actuelle")
SINON
    Trace("L'heure mémorisée est postérieure à l'heure actuelle")
FIN
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Heure(2/2)

2. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
3. Le volet "Trace du débogueur" contient les informations suivantes :

```
12:00:00:00  
12:00:00  
L'heure mémorisée est antérieure à l'heure actuelle
```

- l'affichage en clair de l'heure.
- l'affichage en clair de l'heure selon un masque.

Partie 02 - TOUR D'HORIZON DES VARIABLES

Durée

Pour les durées, le même type de code peut être saisi.

1. Supprimez le code saisi dans l'événement "Initialisation" du projet (Ctrl + A pour tout sélectionner, et Suppr pour supprimer).
2. Saisissez le code suivant :

```
//Déclaration d'une durée
MaDurée est une Durée

//Affectation d'une durée
// - avec une valeur "en clair"
MaDurée = 2 min 8 s
// ou
MaDurée = 128 s
// - en fournissant la durée en secondes, en minutes, ...
MaDurée.EnSecondes = 128

//Affichage en clair de la durée
Trace(MaDurée.VersChaîne("MM:SS"))
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

File(1/4)

Le type File

Le WLangage propose les types File et Pile. Ces types de variables sont des types structurés permettant de regrouper des séries d'éléments.

Pour tester la manipulation des Files, nous allons tester un petit exemple de code :

1. Supprimez le code saisi dans l'événement "Initialisation" du projet (Ctrl + A pour tout sélectionner, et Suppr pour supprimer).
2. Saisissez le code suivant :

```
MaFile est une File de chaînes
```


Partie 02 - TOUR D'HORIZON DES VARIABLES

File(2/4)

Cette ligne de code permet de déclarer une variable de type File contenant des chaînes de caractères.

3. Maintenant, voyons comment ajouter des données. Saisissez le code suivant :

```
//Ajout de données  
//L'élément est ajouté à la fin de la file  
MaFile.Enqueue("WINDEV")  
MaFile.Enqueue("WEBDEV")  
MaFile.Enqueue("WINDEV Mobile")
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

File(3/4)

4. Pour récupérer les données de la File, il suffit d'utiliser la fonction <Variable File>.Défile.

Saisissez le code suivant :

```
// Récupération de la valeur en début de file  
Donnée est une chaîne  
MaFile.Défile(Donnée)  
Trace(Donnée)
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

File(4/4)

5. Les propriétés peuvent également s'appliquer aux variables de type File. Par exemple, la propriété Occurrence permet de connaître le nombre d'éléments de la File.

```
//Comptage  
Trace("La file comporte [%MaFile.Occurrence%]éléments")
```

6. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
7. Le volet "Trace du débogueur" contient les informations suivantes :

```
WINDEV  
La file comporte 2 éléments
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Pile(1/2)

Le même type de code peut être saisi pour tester la manipulation des Piles :

```
//Déclaration d'une pile
MaPile est une Pile de chaînes

//Ajout de données
//L'élément est ajouté au sommet de la pile
MaPile.Empile("WINDEV")
MaPile.Empile("WEBDEV")
MaPile.Empile("WINDEV Mobile")

//Récupération de la valeur du sommet dans une variable
Donnée est une chaîne
MaPile.Dépile(Donnée)
Trace(Donnée)

//Comptage
Trace("La pile comporte [%MaPile.Occurrence%] éléments")
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Pile(2/2)

3. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
4. Le volet "Trace du débogueur" contient les informations suivantes :

```
WINDEV Mobile  
La pile comporte 2 éléments
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Enumération et Combinaison

Les types Enumération et Combinaison

Les énumérations et les combinaisons sont des types de variables qui permettent de manipuler simplement un ensemble de valeurs.

- Une variable de type Enumération peut être affectée uniquement avec une valeur de l'énumération.
- Une variable de type Combinaison peut être affectée avec une ou plusieurs valeurs de la combinaison.

Quel que soit le type de variable manipulé, il faut déclarer la variable, l'affecter avec une ou plusieurs valeurs pour pouvoir ensuite l'utiliser.



```
//Déclaration de l'énumération
```

```
Couleur est une Enumération
```

```
    Vert
```

```
    Bleu
```

```
    Rouge
```

```
    Jaune
```

```
FIN
```

```
//Déclaration d'une variable de type énumération
```

```
CouleurTrait est un Couleur
```

```
//Affectation d'une valeur
```

```
CouleurTrait = Bleu
```

```
//Comparaison de valeur
```

```
SI CouleurTrait = Jaune ALORS
```

```
    Trace("Le trait est jaune")
```

```
SINON
```

```
    Trace("Le trait n'est pas jaune")
```

```
FIN
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Enumération(2/2)

3. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
4. Le volet "Trace du débogueur" contient les informations suivantes :

Le trait n'est pas jaune

Partie 02 - TOUR D'HORIZON DES VARIABLES

Combinaison(1/2)

```
//Déclaration de la combinaison
Options est une Combinaison
    SiègeChauffant
    ToitOuvrant
    SellerieCuir
    BoiteAutomatique
FIN

//Déclaration d'une variable de type combinaison
OptionsVéhicule est un Options

//Affectation d'une valeur
OptionsVéhicule = SiègeChauffant + SellerieCuir + BoiteAutomatique

//Tester si une valeur est activée
SI OptionsVéhicule[SiègeChauffant] = Vrai ALORS
    Trace("L'option siège chauffant est activée")
FIN
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Combinaison(2/2)

3. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
4. Le volet "Trace du débogueur" contient les informations suivantes :

L'option siège chauffant est activée

Partie 02 - TOUR D'HORIZON DES VARIABLES

Structure(1/4)

Le type Structure

- Une structure est un type de données personnalisé. Une structure regroupe des variables de types différents.

1. Pour tester la manipulation des structures, nous allons tester un petit exemple de code :

Supprimez le code saisi dans l'événement "Initialisation" du projet (Ctrl + A pour tout sélectionner, et Suppr pour supprimer).

2. Saisissez le code suivant :

```
NomComplet est une Structure  
    Nom est une chaîne  
    Prénom est une chaîne  
FIN
```

Ce code permet de déclarer une structure composée de deux variables, Nom et Prénom. Les variables composant une structure sont appelées "Membre". Pour manipuler les membres, il faut déclarer une variable du type de la structure.

Partie 02 - TOUR D'HORIZON DES VARIABLES

Structure(2/4)

- Nous allons maintenant :
 3. Déclarer une variable de type structure. Ajoutez le code WLangage suivant.

```
Contact est un NomComple
```

- Affecter les membres de la structure. Ajoutez le code WLangage suivant :

```
Contact.Nom = "POLO"  
Contact.Prénom = "MARCO"
```

- Pour lire le contenu d'un membre, il suffit d'utiliser la syntaxe :

```
<Nom de la variable>.<Nom du membre>
```

Partie 02 - TOUR D'HORIZON DES VARIABLES

Structure(3/4)

- Dans notre exemple, saisissez le code suivant :

```
LeNom est une chaîne  
LeNom = Contact.Nom  
// Affiche le contenu de la variable LeNom  
Trace(LeNom)  
// Affiche directement Contact.Nom (sans passer par une variable)  
Trace(Contact.Nom)
```

4. Testons immédiatement ce code : cliquez sur **GO** parmi les boutons d'accès rapide.
5. Le volet "Trace du débogueur" contient les informations suivantes :

```
POLO  
POLO
```

// Déclaration de la structure

DéfinitionAdresse est une Structure

NomSociété est une chaîne

NomContact est une chaîne

Adresse est une chaîne

CodePostal est une chaîne

Ville est une chaîne

Pays est une chaîne

FIN

// Déclaration d'une variable de type structure

UneAdresse est un DéfinitionAdresse

// Remplissage d'une structure

UneAdresse.NomSociété = "PC SOFT"

UneAdresse.NomContact = "Paul Durant"

UneAdresse.Adresse = "142 Avenue des Champs Élysées"

UneAdresse.CodePostal = "75008"

UneAdresse.Ville = "PARIS"

UneAdresse.Pays = "FRANCE"

// Déclaration d'un tableau de structures

MesAdresses est un tableau de DéfinitionAdresse

Ajoute(MesAdresses, UneAdresse)

Trace(MesAdresses[1].NomSociété)

Partie 02 - LES PROCÉDURES

Une procédure permet d'associer un identificateur (un nom) à une portion de code afin de la réutiliser.

Dans cette leçon, nous allons voir les différents types de procédures existant en WLangage, leur mode de création, comment les appeler, leur passer des paramètres et récupérer un résultat.

1. Ouvrez si nécessaire le projet "WLangage" que vous avez créé dans la première leçon de cette partie
2. Pour afficher les événements WLangage liés au projet :
3. Dans la barre des éléments ouverts, faites un clic droit sur le bouton "P". Le menu contextuel s'affiche.
4. Sélectionnez l'option "Code de l'élément".
5. L'éditeur de code affiche les différents événements associés au projet.
6. Si vous avez suivi les leçons précédentes, supprimez le code saisi dans l'événement "Initialisation" du projet (Ctrl + A pour tout sélectionner, et Suppr pour supprimer).

Partie 02 - LES PROCÉDURES

Une procédure est une suite d'instructions regroupée derrière un nom.

Par exemple, nous allons écrire une procédure qui affiche simplement un message.

1. Nous allons créer une procédure globale dans la collection de procédures du projet WLangage.

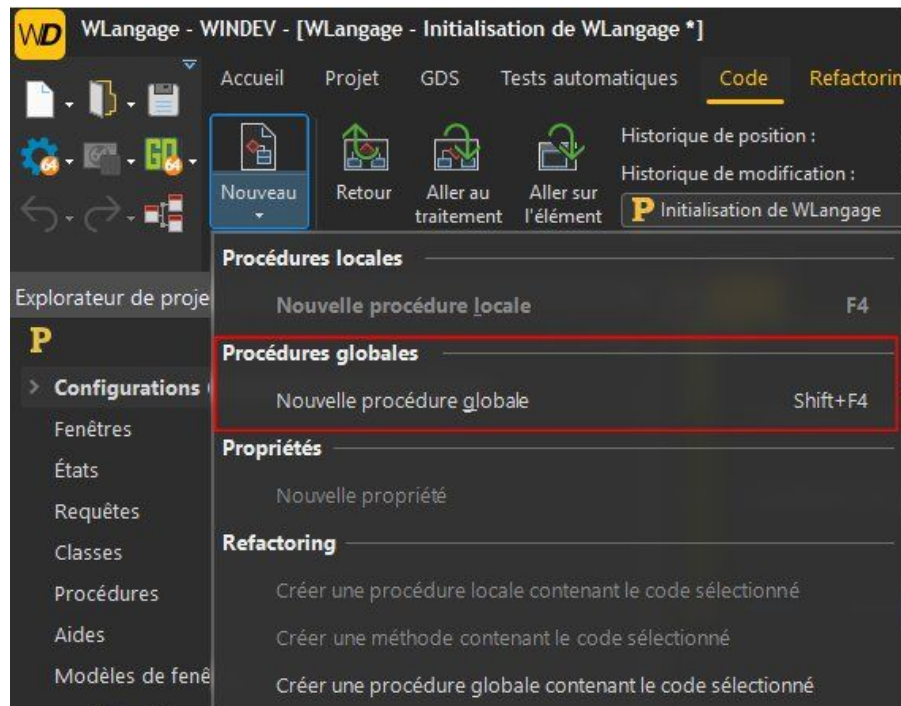


A retenir

Une collection de procédures permet de regrouper toutes les procédures globales à un projet. Ces procédures pourront être appelées depuis n'importe quel code du projet.

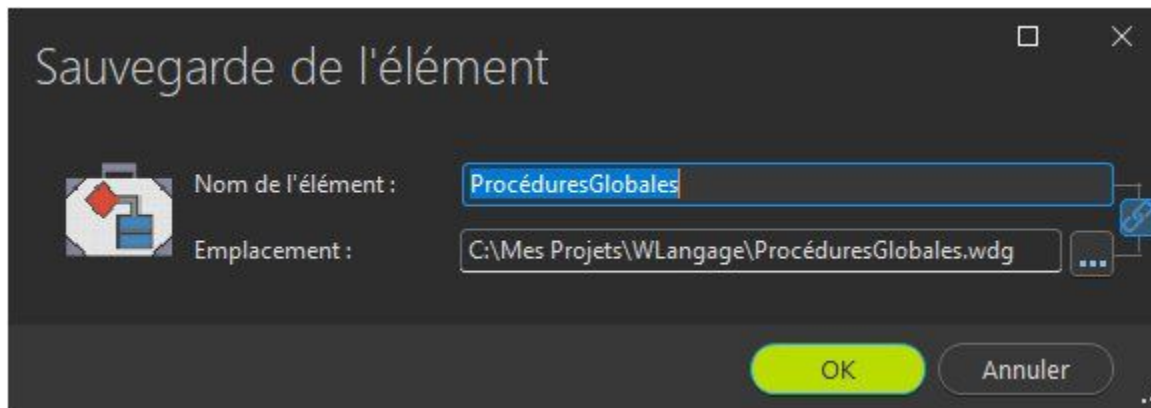
Partie 02 - LES PROCÉDURES

1. Dans le volet "Code" du ruban, dans le groupe "Procédures", déroulez "Nouveau" et sélectionnez "Nouvelle procédure globale".



Partie 02 - LES PROCÉDURES

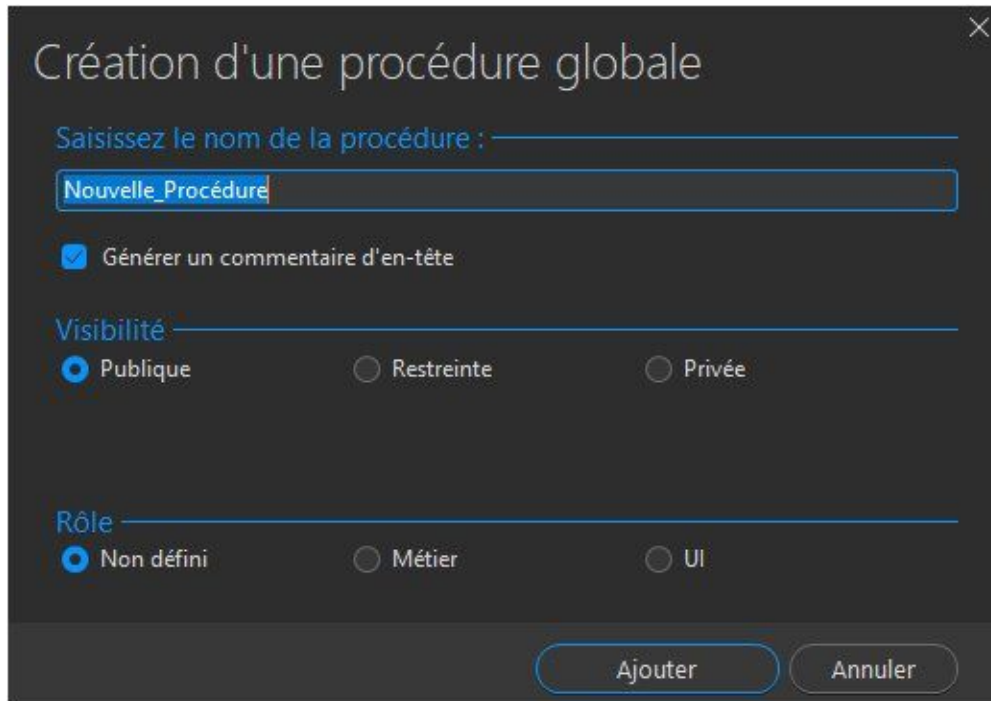
2. Dans la fenêtre de sauvegarde de l'élément qui s'affiche, un nom est proposé par défaut pour la collection de procédures globales.



Conservez ce nom et validez (bouton "OK").

Partie 02 - LES PROCÉDURES

3. Une nouvelle fenêtre s'affiche permettant de saisir le nom de la procédure et ses caractéristiques :



The screenshot shows a dark-themed dialog box titled "Création d'une procédure globale" with a close button (X) in the top right corner. The dialog contains the following elements:

- A label "Saisissez le nom de la procédure :" followed by a text input field containing "Nouvelle_Procédure".
- A checked checkbox labeled "Générer un commentaire d'en-tête".
- A section titled "Visibilité" with three radio button options: "Publique" (selected), "Restreinte", and "Privée".
- A section titled "Rôle" with three radio button options: "Non défini" (selected), "Métier", and "UI".
- At the bottom, there are two buttons: "Ajouter" and "Annuler".

Partie 02 - LES PROCÉDURES

Arrêtons-nous un instant sur cette fenêtre. Il est possible :

d'indiquer le nom de la procédure. Un nom par défaut est proposé et peut être modifié.

de générer un commentaire d'en-tête.

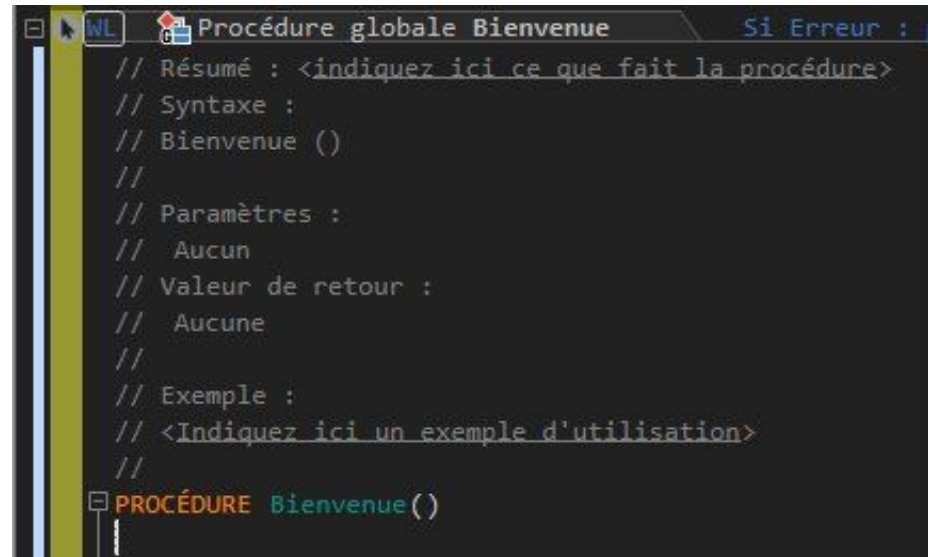
Ce commentaire contient les différentes caractéristiques de la procédure. Le contenu de ce commentaire est par défaut généré au fur et à mesure de la saisie du code de la procédure. Il peut également être modifié directement.

de définir la "portée" de la procédure. Une procédure globale est publique par défaut : elle est accessible depuis n'importe quel code. Dans certains cas, il est nécessaire de restreindre l'accès à une procédure globale. Elle peut être :

- "Restreinte". Dans ce cas, la procédure globale est accessible soit par une autre procédure globale présente dans la collection de procédures soit par une procédure globale d'une autre collection de procédures du projet.
- "Privée". Dans ce cas, la procédure globale est accessible uniquement par une autre procédure globale présente dans la collection de procédures.

Partie 02 - LES PROCÉDURES

3. Pour notre exemple, nous allons modifier uniquement le nom de la procédure et conserver les options par défaut. Indiquez le nom de la procédure "Bienvenue" et validez.
4. La procédure globale "Bienvenue" est affichée sous l'éditeur de code.



```
// Résumé : <indiquez ici ce que fait la procédure>
// Syntaxe :
// Bienvenue ()
//
// Paramètres :
//   Aucun
// Valeur de retour :
//   Aucune
//
// Exemple :
// <Indiquez ici un exemple d'utilisation>
//
PROCÉDURE Bienvenue()
```

Partie 02 - LES PROCÉDURES

Cette procédure est composée :

- de commentaires générés automatiquement.
- du code :

```
PROCÉDURE Bienvenue()
```

Ce code correspond au prototype de la procédure.



A retenir

Les éléments clé d'une procédure sont le mot-clé PROCEDURE suivi du nom de la procédure et de parenthèses.

Partie 02 - LES PROCÉDURES

6. Nous pouvons saisir le contenu de notre procédure. Nous allons faire simple et afficher uniquement un message. Saisissez la ligne de code suivante :

```
Info("Bienvenue dans mon programme écrit en WLanguage")
```

Voilà, notre procédure est finie. Mais pour voir le résultat de cette fonction, il faut l'appeler. C'est notre prochaine étape.

Partie 02 - LES PROCÉDURES

6. Nous pouvons saisir le contenu de notre procédure. Nous allons faire simple et afficher uniquement un message. Saisissez la ligne de code suivante :

```
Info("Bienvenue dans mon programme écrit en WLanguage")
```

Voilà, notre procédure est finie. Mais pour voir le résultat de cette fonction, il faut l'appeler. C'est notre prochaine étape.

EXERCICE 1

1. Écrire un programme WLangage qui permet de d'afficher le jour , le mois et l'année pour la date d'aujourd'hui.

Date d'aujourd'hui est 24/01/2023

Jour : 24

Mois : 01

Année : 2023

2. Afficher la date 05/01/1998 en clair

Mardi 05 Janvier 1998

3. Afficher l'heure actuelle .

EXERCICE 2

10	8	25	1	0	55	12	56	9
----	---	----	---	---	----	----	----	---

Ecrire un programme qui permet de remplir un tableau d'entier nommé nomTab contient les valeurs suivant

1. Afficher le contenu du tableau
2. Calculer et afficher la somme/Produit du tableau
3. Afficher les entiers pairs supérieurs à 10
4. Ajouter les Valeurs 12 | 23 | 88 à la fin du tableau.