Côté cours : enseigner la programmation par l'expérimentation

Propriétés	Description
Intitulé long	Enseigner la programmation par l'expérimentation et l'utilisation d'un environnement de développement.
Formation concernée	Terminale GSI
Matière	Gestion des systèmes d'information
Notions	Partie C, paragraphe 2.3 du programme. Les bases de la programmation : instructions, structures algorithmiques, les tableaux, les fonctions.
Transversalité	
Présentation	Ce cours se propose de montrer comment un environnement de développement peut être utilisé au service des apprentissages de la programmation et de l'algorithmique.
Pré-requis	Les notions vues en première STG.
Outils	VBA
Mots-clés	Programmation, instructions, structures algorithmiques, fonction, tableaux, VBA.
Version	1.0
Auteur(es)	Patrice Grand, Frédérique Parisot, Séverine Quesque

L'apprentissage de la programmation

Le programme de terminale GSI fixe ainsi le cadre d'enseignement de la programmation :

Thèmes	Sens et portée de l'étude	Notions et contenus à construire
2.3. Programmation des traitements	Les traitements constituent la partie dynamique de l'application, ils permettent de mettre en œuvre des règles de gestion. L'adaptation de l'application nécessite d'effectuer des modifications sur le code existant en utilisant les instructions et les bibliothèques de fonctions et de classes fournies par le logiciel de développement. Au cours du développement, la mise au point de l'application nécessite de réaliser des tests et d'avoir recours aux outils de mise au point fournis par le logiciel de développement.	Procédure, fonction, paramètre Structure de contrôle Mise au point de l'application

Les indications complémentaires permettent de mesurer la portée de ces apprentissages :

2.3. Programmation des traitements

Pour réaliser l'adaptation des traitements, l'élève utilise un logiciel de développement et en particulier les éléments prédéfinis : fonctions, procédures et classes d'objets techniques (par exemple les jeux d'enregistrements, les différents contrôles graphiques). L'élève construit des fonctions ou des procédures dans un objectif de modularité ou de réutilisation. La conception et la réalisation de classes sortent du cadre de ce programme : seule l'utilisation d'objets prédéfinis est requise.

La programmation des traitements s'appuie sur une démarche de résolution de problème. Il s'agit notamment pour l'élève :

- de définir les nouveaux résultats à obtenir ;
- d'identifier les données en entrée du traitement et de repérer les structures de données correspondantes ;
- d'identifier les différentes structures de contrôle et les algorithmes types mis en œuvre (exprimés dans le langage de programmation du logiciel de développement) ; par exemple : parcours séquentiel d'un tableau ou d'un jeu d'enregistrements, avec ou sans traitement cumulatif, avec ou sans regroupement ; recherche séquentielle dans un tableau ou un jeu d'enregistrements trié ou non ;

- de proposer les modifications à apporter au programme (modification de la structure du programme, ajout d'instructions et intégration dans la structure).

On peut extraire de ces indications quelques points importants :
- L'enseignement de la programmation doit se construire à partir de besoins clairement identifiés
- L'utilisation d'un environnement de développement (et pas seulement d'un langage) est fortement préconisée
- L'élève doit être placé dans une problématique d'observation et ensuite d'évolution (maintenance) de code.

Le dernier point évoqué nécessite de fournir aux élèves les éléments nécessaires : l'intervention sur du code doit être accompagnée d'apprentissages et savoirs-faire. Ces derniers se construiront également à partir de l'observation, l'interprétation et la modification de code présenté.

Choix didactiques

Les apprentissages proposés dans ce document utilisent le langage VBA dans un environnement de développement « riche ». En effet, il sera fait systématiquement appel à la trace des programmes et l'évaluation pas à pas des variables.
L'utilisation à priori d'un langage ne doit pas affranchir de la présentation d'un modèle : ce modèle s'appuiera sur des schémas.
Le modèle proposé assimilera une instruction à un *mouvement d'information*, un programme à une succession d'instructions et une structure algorithmique à une rupture dans le séquencement.
D'autres choix seront faits également, de moindre portée : pas d'alternative imbriquée, présentation d'une seule structure répétitive (While). Ces derniers choix sont faits ici car il ne s'agit pas de proposer un cours complet mais plutôt de présenter une réflexion et de proposer une démarche. D'autres apprentissages seront à consolider ensuite.

Présentation du support

Le cours proposé ici présente un scénario d'apprentissage de la programmation (à partir du langage VBA) avec pour chaque notion nouvelle des indications et justifications des choix.
Le contexte utilisé est le contexte de « InfoDev » du Certa mais il peut être très facilement mis en œuvre dans un autre contexte.

Contexte

Le fichier Excel de suivi de projets définit des paramètres qui sont ensuite utilisés dans différentes feuilles de calcul. Ainsi, les employés d'InfoDev possèdent par un code d'identification :

AME	Amieau Eline	TECH1
ATN	Ateur Nordine	TECH1
DGR	Gillot Rino	TECH1
LCE	Lacoste Emmanuelle	CPMOE
LFL	Leforestier Loic	COMM
LZJ	Lopez Jésus	TECH2
PKK	Piatsky Karole	TECH1
PNS	Papion Sylvestre	TECH2
PRI	Péret Inès	TECH1
RGT	Regnier Tania	TECH1
VRJ	Vier Janny	TECH3

Évolution envisagée.

On envisage de normaliser l'attribution des codes d'identification et de donner à chaque employé un mot de passe qui sera utilisé pour accéder au réseau de l'entreprise.
Le code sera constitué par la concaténation du nom et de l'initiale du prénom ; ainsi *Amieau Eline* aura pour code d'identification *AmieauE*.

Concernant le mot de passe, il sera construit aléatoirement à partir des 26 lettres majuscules de l'alphabet. Pour des raisons de confusion possible avec zéro et un, les lettres « O » et « I » seront exclues.
Avant d'automatiser le processus deux programmes distincts seront testés. Le langage VBA sera utilisé.
Remarque : à titre indicatif l'objectif final est présenté en annexe sous la forme de deux programmes distincts, *ProgrammeUser* et *ProgrammeMDP*.

Scénario proposé

Avant de mener à bien cette tâche les élèves sont invités à découvrir le langage. Pour cela différents programmes sont fournis ; un questionnement approprié doit permettre d'évaluer les savoirs-faire visés. Les différents programmes proposés sont en relation avec l'objectif final de gestion des identifiants et mot de passe ; c'est pourquoi ils utilisent tous des chaînes de caractères.

A. Les instructions comme flux d'information

A.1 L'affectation

Remarque importante : il est nécessaire d'indiquer la clause *Option Explicit* en haut de la page du module, afin que le langage vérifie que toute variable utilisée a bien été déclarée –ce qui n'est pas le cas par défaut-. Ceci permet de révéler certaines erreurs concernant les identificateurs des variables avant l'exécution.

```
Public Sub Programme1()
   Dim code As String
   Dim debutCode As String
   Dim finCode As String
   debutCode = "Amieau"
   finCode = "Eline"
   code = debutCode & finCode
End Sub
```

Les élèves sont amenés à recopier ce code dans un environnement de développement VBA.

Objectifs visés :
- Savoir utiliser l'environnement : exécuter, utiliser le débogueur (tracer, évaluer le contenu des variables)
- Observer la succession des instructions (séquentialité)
- Modéliser l'affectation
- Découvrir l'opérateur de concaténation (son rôle).

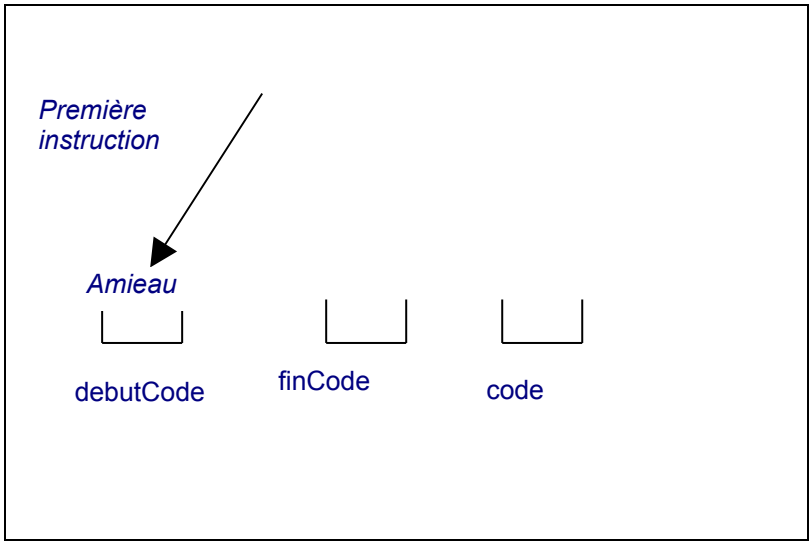
Remarques
- Les élèves ont vu la notion d'affectation et de variable en première STG ; cet exercice vise à en illustrer l'implémentation dans un langage
- Il n'y a pas d'affichage ; ceci afin de montrer d'emblée qu'un programme peut effectuer des traitements sans fournir de sortie écran.

Questions posées :

La trace d'un programme permet de visualiser, après chaque instruction, le contenu des variables.
En utilisant la trace du programme, remplir le tableau suivant :

	Valeurs des variables		
Instructions	code	debutCode	finCode
Avant l'instruction *debutCode = "Amieau"*			
Après l'instruction *debutCode = "Amieau"*			
Après l'instruction *finCode = "Eline"*			
Après l'instruction *code = debutCode & finCode*			

Compléter le schéma suivant en faisant figurer et en ordonnant les mouvements d'information :



Commentaires :

- L'affectation est présentée comme un mouvement d'information au sein de la mémoire ; elle prend deux formes ici : prise de valeur et mouvement entre plusieurs zones.
- La première flèche indique le mouvement en mémoire vers la variable debutCode.

L'apprentissage peut être prolongé par des exercices similaires.

A.2 Sortie d'information à l'écran.

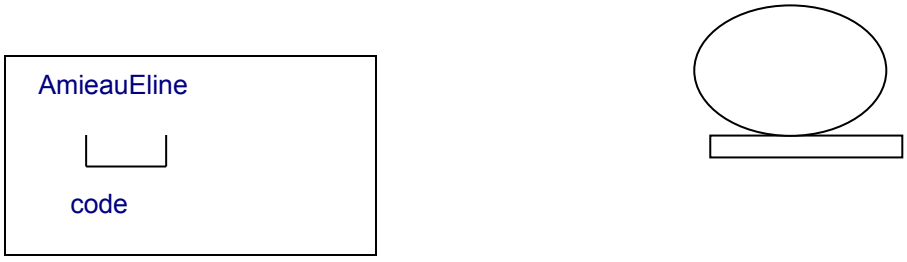
L'exercice 1 est repris avec une nouvelle instruction.

```
Public Sub Programme2()
   Dim code As String
   Dim debutCode As String
   Dim finCode As String
   debutCode = "Amieau"
   finCode = "Eline"
   code = debutCode & finCode
   MsgBox(" valeur du code"  & code)
End Sub
```

Objectif visé.

- Schématiser la sortie écran

On demandera aux élèves d'exécuter le programme puis de compléter le schéma en faisant figurer le flux d'information correspondant à la dernière instruction :



L'apprentissage peut être prolongé par des exercices similaires.

A.3 Entrée d'information au clavier

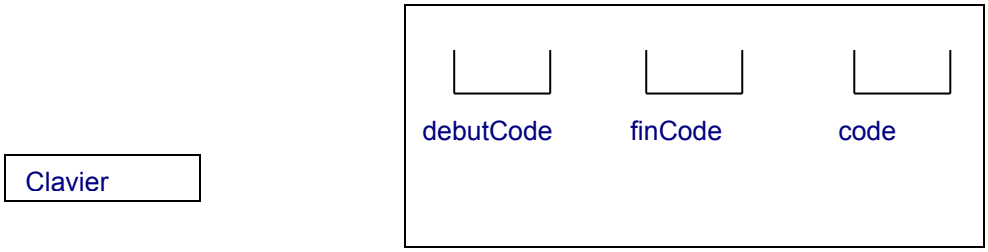
L'exercice 2 est repris en intégrant des entrées d'information :

```
Public Sub Programme3()
   Dim code As String
   Dim debutCode As String
   Dim finCode As String
   debutCode = InputBox(" saisir le début du code")
   finCode = InputBox(" saisir la fin du code")
   code = debutCode & finCode
   MsgBox (" valeur du code : " & code)
End Sub
```

Objectif visé :

- Schématiser une entrée clavier

On demandera aux élèves de tracer le programme puis de compléter le schéma en faisant figurer les flux d'information correspondant aux trois premières instructions :



L'apprentissage peut être prolongé par des exercices similaires.

Remarque :

Le modèle utilisé fait systématiquement référence à des **flux d'entrée et de sortie**, *concrétisés ici par le clavier et l'écran* ; les schémas utilisés doivent permettre de mieux illustrer les propos.

A.4 Utilisation de fonctions

Il ne s'agit pas ici de découvrir l'utilisation de fonctions prédéfinies du langage ; on considèrera que cet apprentissage a été présenté aux élèves. Ce sont donc ici des pré-requis dont on précise le contenu :
Pré-requis :

- Reconnaître dans le code la syntaxe de l'appel d'une fonction
- Interpréter l'aide en ligne du langage (nom, rôle et type des arguments, service rendu de la fonction, valeur de retour)
- Utiliser une fonction par un appel respectant ses spécifications.

A titre d'information, l'atelier C de l'université d'été (http://www.reseaucerta.org/gsi/universite/actes/) propose une démarche de découverte précoce des fonctions (avec un tableur).

L'objectif ici est de réinvestir ces pré-requis dans un contexte particulier.
Le code suivant peut être proposé aux élèves :

```
Public Sub Programme3()
   Dim txt As String
   Dim ch As String
   txt = "Eline"
   ch = Left(txt, 1)
   MsgBox (ch)
End Sub
```

Objectifs :
- Rédiger les spécifications d'une fonction (fonctionnalité, arguments, valeur de retour).
- Modifier des arguments.

Après avoir fait tracer le programme et fait observer l'évolution des variables on peut demander aux élèves de rédiger les spécifications de la fonction *Left* ; ensuite d'extraire les 3 premiers caractères

On peut envisager l'exercice suivant :
Objectifs :
- Interpréter l'aide en ligne du langage en rédigeant les spécifications simplifiées de la fonction Mid
- Mettre en œuvre la fonction dans un contexte

Soit l'aide de la fonction Mid :

Mid(***string***, ***start***[, ***length***])

La syntaxe de la fonction **Mid** comprend les arguments nommés suivants :

Élément	Description
string	Expression de chaîne dont sont extraits les caractères à renvoyer. Si l'argument ***string*** contient une valeur de type Null, **Null** est renvoyé.
start	Donnée de type Long. Position du caractère dans l'argument ***string*** qui marque le début de la partie à extraire. Si la valeur de l'argument ***start*** est supérieure au nombre de caractères contenus dans l'argument ***string***, la fonction **Mid** renvoie une chaîne de longueur nulle ("").
length	Facultatif. Donnée de type **Variant** (**Long**). Nombre de caractères à renvoyer. Si l'argument ***length*** est omis ou si le nombre de caractères du texte (y compris le caractère qui occupe la position ***start***), est inférieur à la valeur de cet argument, tous les caractères à compter de la position ***start*** et jusqu'à la fin de la chaîne sont renvoyés.

Proposition de solution :

Nom de fonction	Service	Arguments	Valeur retournée
MID	Extrait une sous-chaîne	- La chaîne dont on souhaite extraire les caractères - L'indice du premier caractère à extraire - Le nombre de caractères	La sous-chaîne extraite

On peut demander aux élèves de modifier le programme 3 en utilisant la fonction *Mid* à la place de *Left*

B. Les structures algorithmiques

B.1 La structure conditionnelle

La structure conditionnelle sera présentée comme une rupture dans le déroulement séquentiel d'un programme. Le scénario proposé va démarrer par l'observation de l'exécution des instructions du programme, sa trace et la constatation de cette rupture.

Objectif :
- Observer la rupture par des exécutions successives du programme

On présente l'exercice suivant :

```
Public Sub Programme4()
        Dim code As String
        Dim longueur As Integer
        Dim message As String
        code = InputBox(" saisir le code à tester svp")          ' ligne 1
        longueur = Len(code)                                     ' ligne 2
        If longueur >= 2 And longueur <= 5 Then                  ' ligne 3
                Message="Le format du code est valide "                  ' ligne 4
        Else                                                     ' ligne 5
                Message="Le format du code est invalide "        ' ligne 6
        End If                                                   ' ligne 7
        MsgBox (message )                                        ' ligne 8
End Sub                                                          ' ligne 9
```

Remarque :
Il est nécessaire d'être très vigilant sur la présentation des programmes, en particulier sur l'indentation des instructions et structures. Notez ici l'indentation de la structure If.

On peut demander de tracer le programme en fournissant plusieurs valeurs à tester pour la valeur de la variable *code*. Cette observation peut être évaluée à l'aide d'un tableau à compléter faisant apparaître le fonctionnement de la structure alternative :

Valeurs des variables		
code	longueur	Exécution des instructions
« toto »	**?**	Lignes : 1,2, **?, ?, ?…**

code	longueur	Exécution des instructions
« t »	**?**	Lignes : 1,2, **?, ?, ?…**

code	longueur	Exécution des instructions
« tintouin »	**?**	Lignes : 1,2, **?, ?, ?…**

Un second questionnement peut inviter l'élève à modifier le code afin de prendre en compte des contraintes différentes :

- Modifier le programme afin que les codes valides comportent de 1 à 6 lettres au plus
- Modifier le programme afin que les codes comme « toto », « tintouin » soient valides mais pas les codes comme « bob » ou « bobCramer » (entre 4 et 8 lettres)

L'apprentissage peut être prolongé par des exercices similaires et des exercices où les élèves sont amenés à écrire eux-mêmes des tests.

Par contre, il ne semble pas nécessaire de proposer des tests imbriqués dans ce premier apprentissage.

Remarque : le code proposé distingue le traitement :

```
        longueur = Len(code)
        If longueur >= 2 And longueur <= 5 Then
                Message="Le format du code est valide "
        Else
                Message="Le format du code est invalide "
        End If
```

de la sortie d'information (sortie écran ici) :

```
        MsgBox (message )
```

En effet, il aurait été maladroit d'écrire :

```
If longueur >= 2 And longueur <= 5 Then
        MsgBox (" Le format de code est  valide ")
Else
        MsgBox (" Le format de code est invalide ")
End If
```

Car la distinction traitement/sortie n'est pas respectée. Cette distinction est a privilégier, surtout dans les premiers apprentissages ; elle vise à montrer que toutes les parties d'un programme n'ont pas les mêmes fonctionnalités. Un programme prend de l'information d'une source (clavier, base de données…), traite cette information (la modifie ou transforme) et la retourne.

B.2 Une structure répétitive

L'apprentissage peut commencer par l'observation d'un programme qui dénombre une lettre dans un mot :

```
Public Sub programme5()
        Dim mot As String
        Dim longueur As Integer
        Dim compteur As Integer
        Dim lettre As String
        Dim i As Integer
        i = 0
        compteur = 0
        mot = InputBox(" saisir le code à tester svp")
        longueur = Len(mot)
        While i < longueur
                i = i + 1
               lettre = Mid(mot, i, 1)
               If lettre = "e" Then
                   compteur = compteur + 1
               End If
        Wend
       MsgBox (compteur)
End Sub
```

Travail à faire :

- Copier et exécuter ce programme. Saisir le mot « Bateau », relancer en choisissant le mot « marionnette » puis « klaxon »
- Décrire en une phrase ce que fait ce programme.
- En saisissant le mot « terre », compléter le tableau en indiquant les valeurs des variables lorsque la trace est sur la ligne *Wend*

	Valeurs des variables				
	mot	longueur	lettre	i	compteur
Au premier *Wend*	terre				
Au deuxième *Wend*					
Etc…					

Questionnement possible :

- Combien de fois les instructions contenues entre *While* et *Wend* sont-elles exécutées ?
- Dire en une phrase pourquoi.
- A quoi sert la variable i ?
- A quoi sert la variable compteur ?

Evolution possible.
On peut demander de modifier le caractère à rechercher.
On peut ensuite demander de chercher deux caractères (nécessité de deux compteurs distincts).

D'autres exercices peuvent être proposés, analysant des chaînes de caractères. Ainsi on peut montrer un programme qui « nettoie » une chaîne :

```
Public Sub ProgrammeNettoie()
        Dim mot As String
        Dim motNettoye As String
        Dim longueur As Integer
        Dim lettre As String
        Dim i As Integer
        i = 0
        mot = InputBox(" saisir le mot svp")
        motNettoye = ""
        longueur = Len(mot)
        While i < longueur
                i = i + 1
                lettre = Mid(mot, i, 1)
                If lettre <> "-" Then
                        motNettoye = motNettoye & lettre
                End If
        Wend
        MsgBox ("mot nettoyé " & motNettoye)
End Sub
```

Travail à faire :

- Copier et exécuter ce programme. Siasir le mot « Jean-Marc », relancer en choisissant le mot « Jean-de-La-Fontaine »
- Ecrire en une phrase ce que fait ce programme.
- *Tracer* le programme –en saisissant le mot « s-o-s »-et compléter le tableau en indiquant les valeurs des variables lorsque la trace est sur la ligne *Wend*

	Valeurs des variables				
	mot	longueur	lettre	i	motBis
Au premier *Wend*	s-o-s				
Au deuxième *Wend*					
Etc…					

Questionnement possible :

- Combien de fois les instructions contenues entre *While* et *Wend* sont-elles exécutées ?
- Combien de fois l'instruction *motNettoye = motNettoye & lettre* est exécutée ?
- Dire en une phrase pourquoi.

Evolutions possibles :

Nettoyer tous les espaces d'un mot
Nettoyer un mot des caractères « », «_», «-»

C. Structures de données

c.1 Les tableaux, réflexions didactiques

Le tableau représente une structure de données qui permet d'associer une valeur à un indice. Cette notion est très utile en programmation ; d'autre part les langages utilisent le terme de tableau dans des acceptions et des syntaxes assez diverses.
Ainsi en VBA, les indices sont numériques, ordonnés et peuvent commencer à n'importe quelle valeur. En PHP les tableaux peuvent avoir des « indices » qui sont des chaînes de caractères (tableaux associatifs).
L'obstacle pour l'élève est la distinction de trois concepts :

- D'une part, le tableau qui est en soit une donnée (avec un nom, un type, un nombre d'éléments, un type)
- D'autre part ses éléments (type, valeur)
- Et enfin l'indice –qui dans le cas d'un tableau *classique* ne fait pas partie du tableau et n'est qu'un moyen d'accès-.

Ces difficultés se combinent avec les syntaxes peu normalisées des langages : indice du premier élément (1 ou 0), utilisation des parenthèses ou des crochets comme opérateurs d'indexation.

Par ailleurs la déclaration d'un tableau en VBA ne facilite pas sa compréhension :
Dim mots(5) As String
Le type ne réfère pas le tableau comme c'est le cas pour la plupart des autres variables mais bien le type de ses éléments. La dimension du tableau est annoncée par des parenthèses, mais ces parenthèses au moment de la déclaration ne jouent pas le rôle d'opérateur d'indexation. Ainsi l'expression mots(5) prend deux sens selon l'endroit où elle utilisée !
Dim mots(5) As String ' déclare un tableau de 5 String
Mais, par contre
mots(5) retourne la cinquième valeur du tableau, comme par exemple dans l'instruction
MsgBox(mots(5))

Afin de limiter les difficultés liées à la syntaxe en VBA il semble opportun d'adopter la syntaxe suivante :
Dim mots(1 To 5) As String
qui force le premier indice à 1 et qui distingue la syntaxe de déclaration du tableau de celle de l'accès à un élément.

Par ailleurs il faut être vigilant sur le nom donné aux tableaux ; il est préférable de privilégier la sémantique plutôt que la structure de données : *mots* plutôt que *tab*, *notes* plutôt que *tabEntier.* Le *s* indique la multiplicité de la structure.

c.2 Le tableau comme structure de données

Pour observer cette structure de données, il est judicieux d'utiliser le débogueur qui va permettre de visualiser son contenu.

Nous allons utiliser un programme très simple, uniquement pour observer la structure de donnée.

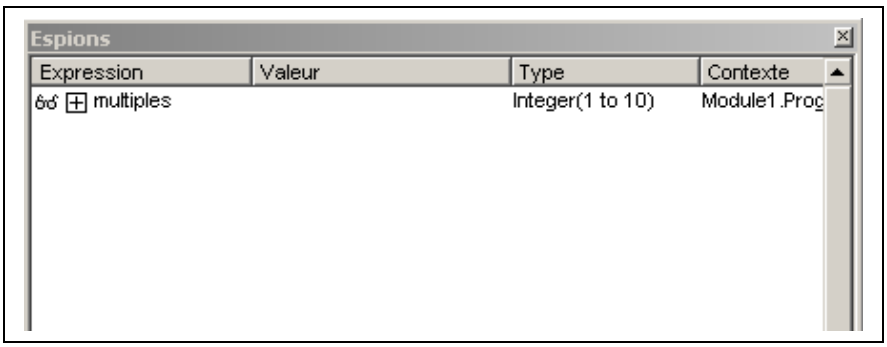
```
Public Sub ProgrammeMultiples()
        Dim multiples(1 To 10) As Integer
        Dim i As Integer
        I = 0
        While i < 10
                i = i + 1
                multiples(i) = 3 * i
        Wend
End Sub
```

Travail à faire

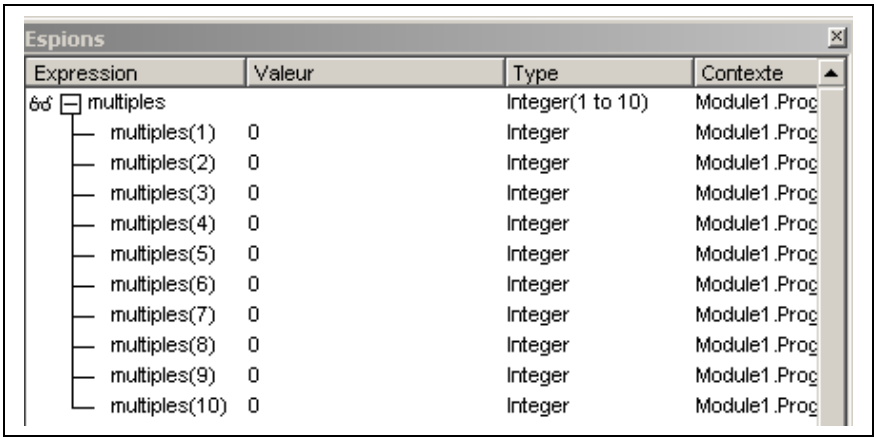
- Tracer le programme
- Aller jusqu'à la ligne indiquée ci-dessous

```
Public Sub ProgrammeMultiples()
    Dim multiples(1 To 10) As Integer
    Dim i As Integer
    i = 0
    While i <= 10
        i = i + 1
        multiples(i) = 3 * i
    Wend
End Sub
```

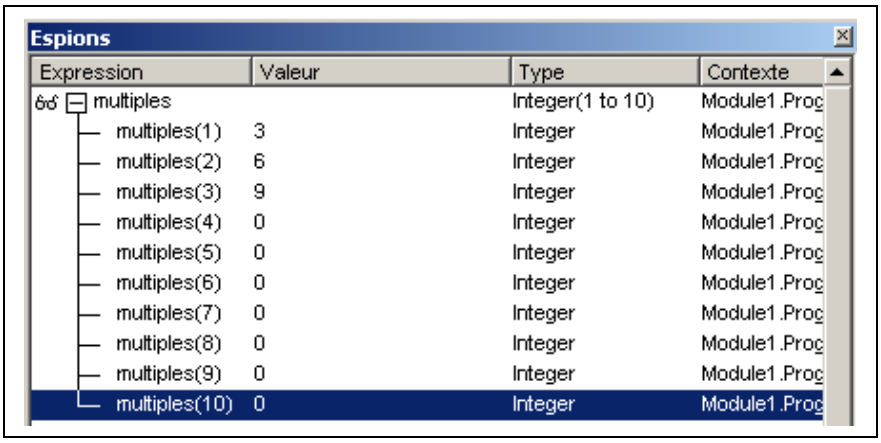
- Placer le curseur sur la variable *multiples* et ajouter un *espion express* (débogage/espion express) , vous devez voir la fenêtre :



- Dépliez la variable *multiples* (en cliquant sur le signe +) afin d'obtenir :



- Continuez à tracer en observant la fenêtre :



Questions

- Quel est le nom du tableau ?
- Quelle est la valeur du 5 ième élément du tableau à la fin de l'exécution du programme
- A quel indice est l'élément de valeur 27 ?

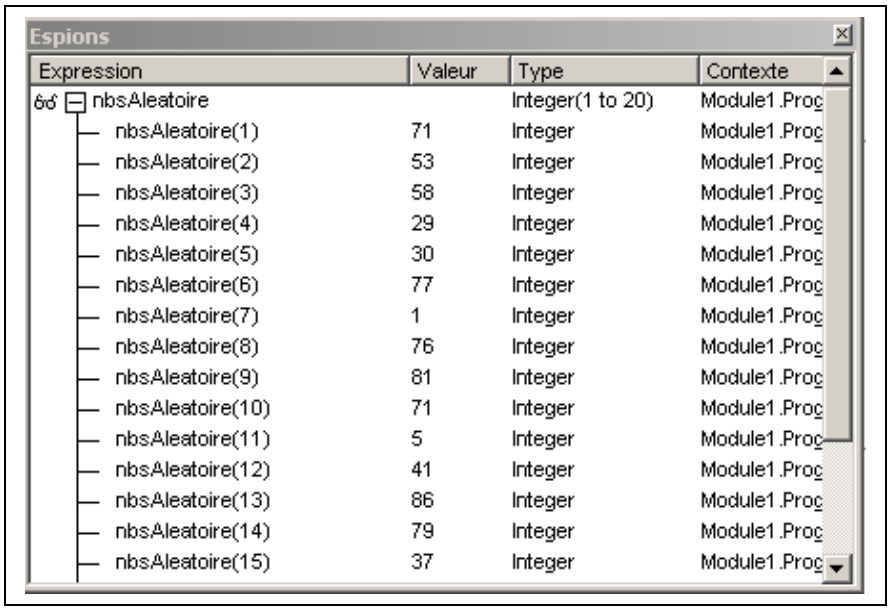
Evolutions

- Modifier la 3 ième valeur du tableau (affecter 11). Vérifier à l'aide de *l'espion*
- Modifier la 7 ième valeur du tableau (affecter 23). Vérifier à l'aide de *l'espion*

On peut proposer ensuite un nouveau programme à observer qui fournit des valeurs moins prévisibles !! Il s'agit d'un générateur de nombres aléatoires

```
Public Sub ProgrammeHasard()
   Dim nbsAleatoires(1 To 20) As Integer
   Dim i As Integer
   i = 0
   While i < 20
      i = i + 1
      nbsAleatoires(i) = (Rnd() * 100)
   Wend
End Sub
```

L'exécution du programme ci-dessus génère un tableau aléatoire pour lequel le questionnement peut être plus mobilisateur :



Les premières questions porteront sur les valeurs et indices, des minimum, des maximum etc…. Ceci afin d'évaluer si les élèves distinguent bien la notion d'indice de celle de valeur.

Remarque : si ce programme est utilisé plusieurs fois il est nécessaire d'ajouter l'instruction *randomize* au début (réinitialisation du compteur utilisé en interne) ; sinon les suites générées seront identiques d'une exécution à l'autre.

D'autres questions peuvent proposer d'intervenir sur le code, par exemple :

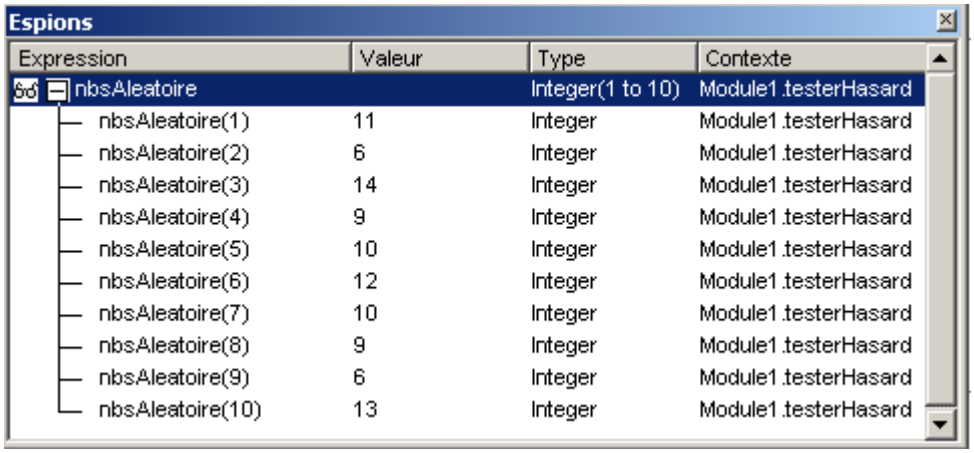
- Quelle est la moyenne (arithmétique) des valeurs
- Combien de valeurs sont inférieures à 50, supérieures à 50

Approfondissement.

Afin d'étudier les performances de la fonction Rnd, on peut demander de construire un tableau de 10 valeurs qui recense le nombre de tirages pour chaque dizaine. Cet exercice est plus délicat et devra être guidé. Voici le code :

```
Public Sub testerHasard()
Dim nbsAleatoire(1 To 10) As Integer
Dim i As Integer
Dim alea As Integer
i = 0
While i < 100
      i = i + 1
      alea = 1 + Int(Rnd() * 10)
      nbsAleatoire(alea) = nbsAleatoire(alea) + 1
Wend
End Sub
```

Un espion express permet de voir le contenu du tableau pour 100 tirages :



Interprétation : il y a eu 11 tirages du nombre aléatoire 1, 6 tirages du nombre aléatoire 2, etc…

D les fonctions et procédures

Au long des apprentissages, l'élève a été amené à utiliser différentes fonctions du langage. Il était placé dans une *posture d'utilisateur de fonctions*, il devait en connaître les spécifications et ainsi respecter leur contrat. Par ailleurs les différents programmes observés ou modifiés, qui sont des procédures, n'ont pas été exploités comme telles, mais comme des programmes indépendants. L'apprentissage va ici se concentrer sur l'écriture de fonctions/procédures qui auront vocations à être *appelées*.

On pourra utiliser un programme déjà étudié afin de mettre en évidence les arguments ; par exemple le programme4 qui permet de valider un code (identifiant) saisi par l'utilisateur :

```
Public Function Valide(unCode As String) As String
        Dim longueur As Integer
        Dim message As String
        longueur = Len(unCode)                                  ' ligne 1
        If longueur >= 2 And longueur <= 5 Then                 ' ligne 2
                message = ""                                    ' ligne 3
        Else                                                    ' ligne 4
                message =   Le format du code est invalide      ' ligne 5
        End If                                                  ' ligne 6
        Valide = message                                        ' ligne 7
End Function
```

Remarques :
On utilise ici une fonction ; ceci permettra éventuellement de l'utiliser dans Excel pour valider une cellule.
Notez l'identificateur utilisé pour l'argument –*unCode*- ; ceci permet d'en préciser déjà la sémantique : il s'agit *d'un quelconque code*. Il est préférable de procéder ainsi et de nommer les arguments formels (ceux qui figurent dans les parenthèses) par un identificateur préfixé d'un article indéfini.

On désire utiliser cette fonction de manière automatique à chaque saisie d'un code dans une cellule par exemple.
Un programme d'appel doit être utilisé :

```
Public Sub AppelValide()
        Dim code As String
        Dim resultat As String
        code = InputBox(" saisir le code à tester svp")         ' ligne A
        resultat = Valide(code)                                 ' ligne B
        MsgBox (resultat)                                       ' ligne C
End Sub
```

Le débogueur sera ici encore privilégié afin de faire découvrir le mécanisme des fonctions.

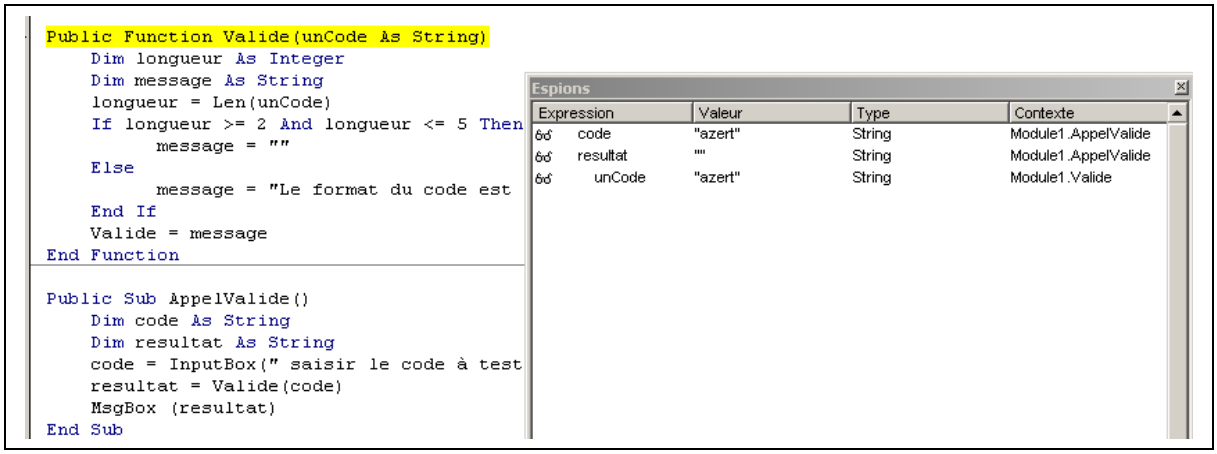
Objectif 1 : interpréter par la trace le branchement à une fonction.
On peut demander de tracer le programme et de reproduire sur papier l'ordre d'exécution des instructions en utilisant les numérotages des lignes.

Objectif 2 : interpréter par la trace le passage d'argument.

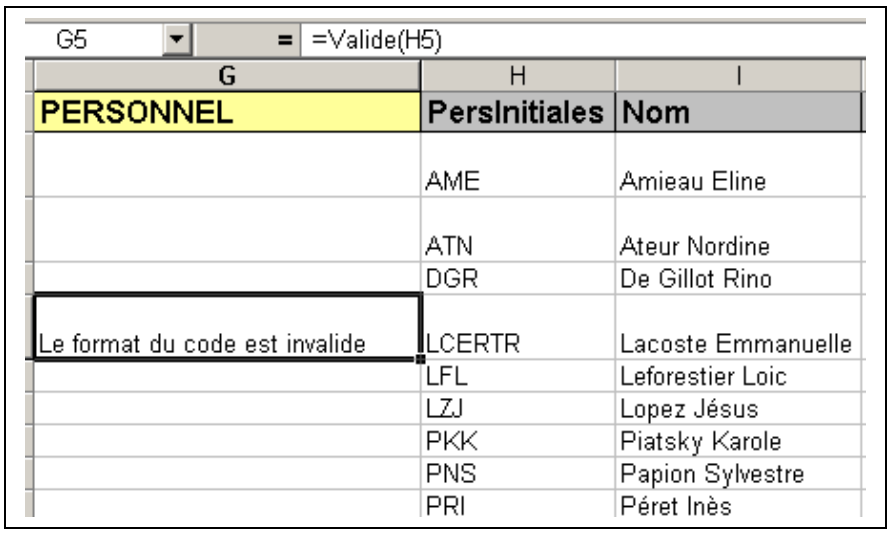
On peut demander de remplir un tableau où figureront plusieurs variables :

	Valeurs des variables		
Variables	code	résultat	unCode
Ligne A			
Ligne B			
Ligne 1			
Ligne 2			
etc			

L'élève sera encouragé à placer des espions pour remplir le tableau.



On pourra ensuite utiliser la fonction Valide dans Excel afin de montrer l'intérêt des fonctions pour la réutilisation du code :



Ce premier apprentissage en appelle bien sûr d'autres qui permettront de consolider les connaissances. On peut envisager de demander de paramétrer la fonction *Valide* par deux arguments qui sont les valeurs d'encadrement ; la signature de la fonction deviendrait :

Public Function Valide(unCode As String unMin As Integer, unMax As Integer) As String

Il est possible aussi de reprendre le programme 5

Conclusion

Ce cours n'est pas exhaustif, il permet d'illustrer l'utilisation d'un environnement de développement dans les apprentissages. D'autres exercices sont nécessaires dans lesquels l'élève sera un peu moins guidé et plus dans une posture de « producteur » de code, voir le programme figurant en annexe et présenté en début de cours.

ANNEXE

Programmes à réaliser en fin d'apprentissage :
1. Construction du code utilisateur :

```
Public Sub ProgrammeUser()
   Dim nomPrenom As String
   Dim nom As String
   Dim prenom As String
   Dim utilisateur As String
   Dim mots(1 To 2) As String
   nomPrenom = InputBox("Taper le nom et le prénom svp")
   mots = Split(nomPrenom)
   mots = mots(1)
   prenom = mots(2)
   utilisateur = nom & Left(prenom, 1)
   MsgBox (utilisateur)
End Sub
```

2. Construction du mot de passe :

```
Public Sub MdP()
   Dim motDePasse As String
   Dim i As Integer
   Dim lettre As String
   Dim nbAleatoire As Integer
   motDePasse = ""
   i = 0
   While i < 4
     nbAleatoire = Int(Rnd() * 25)
     lettre = Chr(65 + nbAleatoire)
     If lettre <> "O" And lettre <> "I" Then
         motDePasse = motDePasse & lettre
         i = i + 1
      End If
   Wend
   MsgBox (motDePasse)
 End Sub
```