COURS Z11 N° 1

Introduction à la programmation

Sommaire

Exemples de projets d'étudiants de Z11 96/97

Généralités

- Communiquer ⇨ Langage (parlé, écrit, par signes …)
- **Langage** : ensemble de **caractères**, de **symboles** et de **règles** permettant de les assembler, dans le but de communiquer.
- **Langages naturels** : celui des hommes, des animaux, des …
- **Langages artificiels** : utilisés pour simplifier la communication (pictogrammes) et surtout en **programmation informatique**.

  Ils sont formellement définis par des "grammaires" (**INF Z18** : Informatique et linguistique)

  Ils sont **interprétables** ou **compilables**.

- **Programmer** : écrire dans un langage de programmation informatique une suite d'instructions, organisée en algorithme dans un but précis, exécutable par un ordinateur.

Revenir au sommaire

Langages de programmation

- **Langage machine**

  Le seul compréhensible par la machine. Assemblage de **0** et de **1** (**bits**).

  Complexe à mettre en œuvre. Domaine de spécialistes.

  Le **langage d'assemblage** (ou Assembleur) permet de développer des programmes proches

des instructions de base d'un microprocesseur.

Exemple :

```
Message DB 'Bonjour' , '$' ; met la chaîne dans une zône mémoire

          MOV AH,09h ; charge le registre A pour afficher

          MOV DX,OFFSET Message ; charge la chaîne dans registre DX

          INT 21h ; appel interruption 21 (affichage)
```

Ligne N° 2 en hexadécimal : **B4 09**

Ligne N° 2 en binaire : **01011 0100 0000 1001**

- **Langages évolués**

  Le programmeur écrit des lignes d'instructions proches du langage naturel.

  Ce **code source** est ensuite soit exécuté ligne à ligne par un **interpréteur** soit traduit en langage machine par un **compilateur** avant l'exécution.

- **LISP** : programmation fonctionnelle.
- **PROLOG** : programmation logique.
- **PASCAL** : programmation procédurale. **(Begin Write ('Bonjour') End.)**
- **C et C++** : programmation de logiciels.
- **VISUAL BASIC** : programmation graphique événementielle.
- **JAVA** : récent, portable, voisin du **C**$^{++}$
- Les 3 derniers sont des langages **objets** ou **orientés objets**.

Revenir au sommaire

Méthode de programmation

- Spécification des besoins des futurs utilisateurs.
- Spécifications fonctionnelles : comment satisfaire aux besoins.
- Conception générale : division du logiciel en programmes.
- Conception détaillée : algorithme le plus adapté pour chaque programme.
- Assemblage des différents programmes.
- Codage à l'aide du langage le plus adapté.
- Validation et qualification.

La **conception** est beaucoup plus importante que le **codage** qui peut être sous-traité dans le cas de gros logiciels.

Revenir au sommaire

Caractéristiques du langage

- Ancien **BASIC** (**B**eginner's **A**ll purpose **S**ymbolic **I**nstruction **C**ode)
- Programmation par **objets** (briques logicielles)
- Programmation **graphique** (fenêtres, icônes, menus, souris…)
- Programmation **événementielle** (sollicitations : souris, clavier, autre événement…)
- Réutilisable (modules de code BASIC).

Revenir au sommaire

Structure générale

- Les objets manipulés sont appelés des **contrôles** (bouton de commande, boîte de dialogue, zône de texte, zône d'image, etc.
- L'interface utilisateur créée est fenêtrée. Une fenêtre est appelée une **feuille** (Form). Une feuille est elle-même un contrôle. Au lancement ⇨ feuille de démarrage.
- Chaque contrôle peut réagir à des **événements** qui lancent des suites d'instructions codées en BASIC.
- Des **modules généraux** de code BASIC peuvent porter sur tout le programme. Ces modules sont réutilisables.

Revenir au sommaire

Contrôles et propriétés

Un objet (contrôle) peut posséder un grand nombre de propriétés par exemple sur sa forme, sa couleur, sa position dans la feuille, sa visibilité, etc.

La plus importante est la propriété **Name** qui donne un nom au contrôle. Ce nom permet de référencer le contrôle.

Syntaxiquement le nom d'un objet est séparé de la propriété par un point

objet . propriété = valeur

Exemple 1 : `Etiquette.CouleurDeFond = Bleu`

Exemple 2 : `Affichage.Caption = "Bonjour"`

Exemple 3 : `Image.Visible = True`

Exemple 4 : `cmdQuitter.Enabled = False`

Revenir au sommaire

Contrôles et événements

Le code d'un événement associé à un contrôle forme une **procédure** événementielle dont la syntaxe générale est :

```
Sub NomContrôle_Evénement()

     Instruction 1

     Instruction 2

     ..............

End Sub
```

Exemple :

```
Sub cmdQuitter_Click()

     Unload Me

     End

End Sub
```

Revenir au sommaire

Interface de développement de VISUAL BASIC

Cette étude sera assurée en "**live**".

Lancer **VB 5.0**

Revenir au sommaire

COURS Z11 N° 2

De quoi débuter

Sommaire

Exemples de projets d'étudiants de Z11 96/97

Principaux objets et propriétés essentielles

- **Form** (**feuille**). C'est le conteneur graphique des contrôles de l'application.

  Propriétés : **Name**, **Caption**, **Picture**

- **CommandButton** (**bouton de commande**)

  Utilisation : exécute le code associé à l'événement **click** sur ce bouton.

  Propriétés : **Name**, **Caption**

- **Label** (**étiquette**)

  Utilisation : affiche une **sortie écran** (texte, nombre, date…) non interactive.

  Propriétés : **Name**, **Caption**, **BackColor**, **BorderStyle**, **Font**, **Alignment**

- **Image** (**image**)

  Utilisation : affiche des images en **mode point** (BitMap au format **.BMP**, **.WMF**, **.ICO**) Peut servir de bouton de commande (événement click).

  Propriétés : **Name**, **Stretch**, **BorderStyle**, **Visible**

- **TextBox** (**zone de texte**)

  Utilisation : pour **taper** ou **afficher** du texte.

  Propriétés : **Name**, **BackColor**, **BorderStyle**, **Font**

  Propriétés fixées au départ (valeurs standard). Modifiées par programmation objet.

Revenir au sommaire

De bonnes habitudes

Dès qu'un objet est créé sur une feuille lui donner un nom significatif (propriété **Name**)

Conseil N° 1 :

- N'utiliser que des lettres et des chiffres (exclure **. - / _** "**espace**" etc.)
- Majuscules initiales des différents mots
- Préfixer selon la nomenclature suivante :

Une **Form** est préfixée par **frm**

Un **CommandButton** est préfixé par **cmd**

Un **Label** est préfixé par **lbl**

Une **Image** est préfixée par **img**

Une **TextBox** est préfixée par **txt** etc.

Exemples : lblAffichage, cmdEntréeDesDonnées, imgFondEcran, frmFeuilleAccueil etc.

Conseil N° 2 :

Dresser une table des propriétés importantes :

Objet	Name	Caption	Autres propriétés
Bouton	cmdQuitter	Fin	
Etiquette	lblSortieRésultat	(vide)	BackColor, Font…
Feuille	FrmJeuDuLoto	Jeu du LOTO	BackColor, Picture…

Revenir au sommaire

Développement d'un projet

Rappel du cours précédent

Spécification des besoins des futurs utilisateurs.
Spécifications fonctionnelles : comment satisfaire aux besoins.
Conception générale : division du logiciel en programmes.
Conception détaillée : algorithme le plus adapté pour chaque programme.
Assemblage des différents programmes.
Codage à l'aide du langage le plus adapté.
Validation et qualification.

Du point de vue technique :

- Préparer un **dossier** qui contiendra l'ensemble du projet (plusieurs fichiers).
- Dessin de **l'interface graphique** selon les fonctionnalités désirées.
- Projet multi-fenêtres ⇨ fixer la **feuille de démarrage** ou procédure **Main().**
- Dresser la table des propriétés essentielles.
- Enregistrer dans le dossier prévu à cet effet.

Revenir au sommaire

Implantation du code

Les objets réagissent à divers événements : Click, MouseUp, MouseDown, Change etc.

Pour implanter le **code** (suite d'instructions produisant des actions pré-définies) lié à un événement il suffit d'opérer un double-clic sur l'objet qui déclenche cet événement.

Exemples :

```
Private Sub cmdQuitter_Click()

     End

End Sub


Private Sub cmdMonNom_Click()

     LblAffichage.Caption = "Mon nom est Personne."

End Sub
```

Dans une programmation plus avancée on pourra écrire du code non lié à des événements dans un module BASIC qui pourra être réutilisé pour un autre projet.

Revenir au sommaire

Exécution pour validation

Le mode **création** permet de construire le projet (ou application).

Le mode **exécution** permet de vérifier si les fonctionnalités prévues sont convenables.

Pour lancer l'exécution : soit le bouton ▶ soit menu Exécution - Exécuter (**F5**).

Pour arrêter l'exécution : soit le bouton ■ soit menu Exécution - Fin.

N.B.

Ne pas oublier d'arrêter un programme sinon toute correction est impossible (notamment pour un programme qui boucle…)

Revenir au sommaire

COURS Z11 N° 3

Interactivité, variables et conditions

Sommaire

Boîte de dialogue - message

Dans un programme l'utilisateur et le concepteur dialoguent par l'intermédiaire de différents canaux (visuels, sonores) à l'aide de **messages** interactifs ou non.

On a l'habitude d'utiliser ces boîtes de dialogue dans des logiciels connus : boîte de connexion réseau, Enregistrer, Imprimer etc.

Pour afficher un message non interactif on utilise la boîte de dialogue prédéfinie **MsgBox**.

Exemple :

```
MsgBox("Salut tout le monde !")
```

C'est un simple message affiché dans une boîte agrémentée d'un bouton **OK**.

En fait c'est plutôt une fonction dont la syntaxe est beaucoup plus complexe (voir l'aide en ligne de VB - menu **?** à la droite de la barre des menus).

Revenir au sommaire

Notion de variable

Elles sont nécessaires pour **stocker** (conserver) une valeur dynamique et réutilisable.

C'est en fait une simple **zone mémoire** qui porte un nom choisi par le programmeur pour faciliter sa programmation. Le nom de la variable est une **adresse mémoire**.

Si l'on veut une programmation cohérente il faut les déclarer avec leur type.

Menu **Outils** - **Options** - onglet **Environnement** - *choisir* : "Requiert la déclaration des variables".

On peut aussi écrire la directive **Option Explicit** au début de la section des déclarations d'un module.

Syntaxe :

```
Dim <NomVariable> As <Type>
```

Pour la lisibilité du code on peut les commenter après une apostrophe (')

Exemples :

```
Dim Taux As Single ' Taux de la TVA

Dim Réponse As String ' Mot proposé par le joueur
```

Pour éviter tout problème il est préférable d'initialiser les variables déclarées.

```
Compteur = 0

Taux = 20,6
```

Le langage Basic utilise **7 types** de données dont les plus utilisés sont le type **String** (chaîne de caractères), le type **Integer** (entier relatif) et le type **Single** (décimal).

Revenir au sommaire

Portée d'une variable

Si une variable est déclarée au début de la procédure qui la manipule (**Dim** ou **Private**) elle n'est alors valide que pour cette procédure. L'existence et la valeur de la variable disparaissent avec l'instruction **End Sub**.

Toute référence à cette variable en dehors de cette procédure provoquera une erreur de compilation.

Si une variable est déclarée dans la section des déclarations d'un module elle est valide dans toutes les procédures du module.

Une variable peut aussi être déclarée **Public** ou **Global** et sera alors valide pour toute l'application.

Exemple :

```
Global MotInitial As String ' premier mot à traiter
```

Revenir au sommaire

Entrée des données

Si l'utilisateur fournit une donnée il faut la stocker dans une **variable** pour pouvoir la réutiliser autant de fois qu'on le veut.

Le plus simple est d'utiliser la boîte de dialogue prédéfinie **InputBox** qui est aussi une fonction et qui retourne une valeur de type **Variant**. Cette fonction a pour effet d'**affecter** une

valeur à une variable dûment déclarée.

Exemple :

```
Valeur = InputBox("Entrez votre donnée ?")
```

Alors la variable **Valeur** contient une donnée du même type que sa déclaration (String, Integer etc.) ou du type de la valeur entrée… (à éviter…)

```
NombreProposé = InputBox("Quelle est votre proposition ?")
```

Alors la variable **NombreProposé** contient un Integer si elle a été déclarée Integer.

Sa syntaxe complète est aussi assez délicate (voir l'aide en ligne de VB).

Revenir au sommaire

Structure conditionnelle

Les instructions à exécuter peuvent dépendre d'une condition. Il faut alors utiliser une structure décisionnelle qui oriente le déroulement du programme vers des blocs d'instructions déterminés. C'est la structure **If … Then … Else … End If**

1. Sur une seule ligne :

```
If condition Then instruction1 [Else instruction2]
```

Condition est une expression dont la valeur est booléenne (**True** ou **False**).

Si cette expression est une valeur numérique, la valeur 0 (zéro) correspond à **False** et toute autre valeur correspond à **True**.

Exemple :

```
If Moyenne >= 10 Then Décision = "Admis" Else Décision = "Refusé"
```

2. Sous forme de bloc :

```
If condition Then
     [instructions]
Else
     [instructions]
End If
```

Exemple :

```
If Moyenne >= 10 Then
```

```
            Admis = Admis + 1
            MsgBox("Candidat admis")
        Else
            Ajournés = Ajournés + 1
            MsgBox("Candidat ajourné")
        End If
```

Revenir au sommaire

COURS Z11 N° 4

Les structures conditionnelles

Sommaire

Nécessité de ces structures

Le programmeur est très souvent amené à **tester** des valeurs et à orienter le programme selon ces valeurs.

L'utilisateur est parfois invité à faire des **choix** que le programme doit prendre en compte.

Plusieurs **structures décisionnelles** permettent ces traitements.

Il s'agit d'utiliser la structure la plus adaptée au problème à résoudre.

Exemples :

- `Traitement d'un mot de passe.`
- `Essai de la part d'un joueur.`
- `Comptage des bonnes réponses.`
- `Choix d'un niveau de jeu.`

Revenir au sommaire

Structure If … Then … Else …

Cette structure décisionnelle revêt 2 formes :

1. Sur une seule ligne :

   Syntaxe : `If condition Then instruction1 [Else instruction2]`

   `Condition` est une expression dont la valeur est booléenne (**True** ou **False**).

   Si cette expression est une valeur numérique, la valeur 0 (zéro) correspond à **False** et toute autre valeur correspond à **True**.

   Exemple : `If MotDePasse <> "zizou" Then End 'assez brutal!`

2. Sous forme de bloc :

Syntaxe :
```
If condition Then
        [instructions]
    Else
        [instructions]
End If
```

Exemple :
```
If NombreProposé > NombreATrouver Then
    MsgBox("Votre nombre est trop grand !")
    Essai = Essai + 1
Else
    MsgBox("Votre nombre est trop petit !")
    Essai = Essai + 1
End If
' Essayez de trouver ce qui manque dans ce test !
```

Revenir au sommaire

Insuffisance de cette structure

Cette structure n'est vraiment valable que pour une alternative (2 possibilités).

Pour traiter 3 possibilités il faudra imbriquer un autre `If` après l'instruction `Else`.

Exemple :
```
If NombreProposé > NombreATrouver Then
    MsgBox("Votre nombre est trop grand !")
Else
    If NombreProposé < NombreATrouver Then
        MsgBox("Votre nombre est trop petit !")
    Else
        MsgBox("Gagné !")
End If
```

```
End If

' Essayez de trouver ce qui manque cette fois !
```

Au-delà de 3 possibilités on a besoin d'une autre structure qui peut gérer **plusieurs cas**.

Revenir au sommaire

Structure Select … Case …

C'est une extension du If … Then … Else … Elle permet une programmation plus claire en évitant une trop grande imbrication de **If** successifs.

Syntaxe :

Select Case Expression

Case **ListeValeurs1**

[*Instructions*]

Case **ListeValeurs2**

[*Instructions*]

[Case Else

Instructions]

End Select

ListeValeurs peut être :

- une suite de valeurs : 1, 3, 5, 7, 9

- une fourchette : 0 To 9

- une plage : **Is** >= 10

Une seule Expression (ou une simple variable) est testée au début puis est comparée avec les listes de valeurs.

A la **première concordance** les instructions correspondantes sont exécutées puis le programme sort de la structure.

Si aucune concordance n'est trouvée les instructions placées après le Else sont exécutées.

Exemple :

```
Select Case CodeASCIICaractère

Case 65, 69, 73, 79, 85

MsgBox(" C'est une voyelle ")
```

```
Case 66 To 90

MsgBox(" C'est une consonne ")

Case Else

MsgBox(" Ce n'est pas une lettre ")

End Select

    ' Essayez de trouver ce qui ne va pas dans ce test !
```

Revenir au sommaire

Structure Iif

C'est exactement la fonction **IF** d'EXCEL.

Syntaxe : **IIf (**Condition**,** ValeurSiVrai**,** ValeurSiFaux**)**

Exemple :

```
Dim Note As Single

Dim Réponse As String

Note = InputBox (" Tapez votre note ")

Réponse = IIf (Note >= 10, " Admis ", " Ajourné ")

MsgBox (Réponse)
```

Il existe d'autres outils décisionnels (**Switch**, **With**, **Choose** ...) qui ne sont pas nécessaires pour un module de niveau 1.

Revenir au sommaire

COURS Z11 N° 5

Tableaux et boucles

Sommaire

Structure de tableau

On a souvent besoin de travailler sur un ensemble de données.

Un exemple géographique : les températures moyennes des 12 mois de l'année.

On pourrait déclarer **12 variables identiques** :

```
Dim Temp1, Temp2, Temp3, Temp4, … … …, Temp12 as Single
```

On dispose d'une structure de données appelée **Tableau** qui permet de conserver dans une seule "**entité**" plusieurs valeurs de même type.

Le nom du tableau est une variable qu'il est recommandé de préfixer par Tab.

Le nombre de valeurs de types identiques est à déclarer entre parenthèses.

Exemple 1 : `Dim TabTemp(12) As Single`

Numéro	1	2	3	4	5	…
Température	6	5,5	7	11,5	15	…

L'accés à la case numéro 3 se fait par `TabTemp(3)` qui vaut 7.

Exemple 2 : `Dim TabMajuscules(65 to 90) As String`

Numéro	65	66	67	…	89	90
Majuscule	A	B	C	…	Y	Z

Revenir au sommaire

Boucles en nombre défini

Cette boucle est utilisée si l'on connait à l'avance le nombre de fois qu'elle sera parcourue.

Syntaxe :

```
For Compteur = Début To Fin [Step Incrément]

Instructions

[ ... Exit For]

[Instructions]

Next [Compteur]
```

Le test est effectué **au début** de la boucle.

La variable numérique Compteur est **incrémentée** à chaque fin de boucle du nombre indiqué par l'incrément. Si l'incrément n'est pas spécifié il est fixé à **1**.

Si la valeur de **Fin** est inférieure à la valeur de **Début** l'incrément est négatif.

La valeur de **Compteur** peut être utilisée (par exemple pour numéroter le passage dans la boucle) mais ne doit pas être modifiée dans le corps de la boucle.

Exemple :

```
For i = 1 To 50

TabInitial(i) = 0 ' Initialisation de chaque case à 0

Next i
```

Revenir au sommaire

Remplissage d'un tableau

Pour remplir un tableau on le balaye avec une boucle For … To … Next (car le nombre de cases est connu à l'avance).

Exemple 1 :

```
Dim TabTemp(12) As Single

Dim Compteur As Integer

For Compteur = 1 To 12

          TabTemp(Compteur)=InputBox("Température N° " & Compteur)
```

```
Next Compteur
```

Exemple 2 :

```
Dim TabTirageLoto(6) As Integer

Dim Compteur As Integer

For Compteur = 1 To 6

          TabTirageLoto (Compteur)=Rnd * 48 + 1

Next Compteur

                    ' Il y a une imperfection dans ce tirage des 6 numéros du Loto !
```

Revenir au sommaire

Traitement des valeurs d'un tableau

On suppose rentrées les 12 températures dans un tableau de Single appelé **TabTemp**.

On veut rechercher la température maximale dans ce tableau de 12 températures.

Il s'agit donc de balayer ce tableau et de conserver la valeur maximale dans une variable.

Au départ on suppose que la température maximale est la première du tableau.

```
Dim Compteur As Integer

Dim TempMaxi As Single

          TempMaxi=TabTemp(1)

For Compteur = 2 To 12

If TabTemp(Compteur)>TempMaxi Then TempMaxi=TabTemp(Compteur)

Next Compteur
```

A la fin du processus la variable **TempMaxi** contiendra la valeur recherchée.

Revenir au sommaire

Déclarations publiques et procédures publiques

Si une variable est déclarée au début d'une procédure événementielle par l'instruction Dim elle n'est alors valide que pour cette procédure. L'existence et la valeur de la variable disparaissent avec l'instruction End Sub.

Toute référence à cette variable en dehors de cette procédure événementielle provoquera une **erreur de compilation**.

Si une variable est déclarée **avant** toute procédure événementielle par l'instruction Public elle est valide dans toutes les procédures de la feuille.

Exemple :

```
Public NomDuJoueur As String ‘ cette variable sera utilisable

     ' dans toutes les procédures événementielles de la feuille.
```

La déclaration d'un tableau ne "supporte" pas le mot réservé Public. Il faudra alors utiliser le mot réservé Dim même si c'est une déclaration publique.

Exemple :

```
Dim TabNotes(5) As Single ‘ Ce tableau sera utilisable dans

               'toutes les procédures événementielles de la feuille.
```

Si une procédure non-événementielle est déclarée **avant** toute procédure événementielle par l'instruction Public elle sera utilisable dans toutes les proc. ev. de la feuille.

Exemple :

```
Public Sub SaisieTempérature()

Température = InputBox("Taper une température")

Somme = Somme + Température

End Sub

'Cela présuppose une déclaration initiale des variables publiques

'Température et Somme.
```

Cette procédure sera exécutée à chaque appel par son nom : `SaisieTempérature`.

Revenir au sommaire

COURS Z11 N° 6

Les structures de contrôle

Sommaire

Présentation du problème

Si le programme doit exécuter un bloc d'instructions en nombre prédéfini on utilise la boucle `For … To … Next.`

Exemple :

```
For i = 1 To 49
                    TabLoto(i) = i 'chaque case contient son numéro
Next i
```

Si le nombre de passages dans la boucle est inconnu au départ, mais dépend d'une **condition** dont la réalisation est imprévisible cette structure n'est pas adaptée.

Exemple 1 :

Demander le mot de passe **tant que** la réponse n'est pas le bon mot de passe.

Demander le mot de passe **jusqu'à ce que** la réponse soit le bon mot de passe.

Exemple 2 :

Demander la saisie d'une note **tant que** la réponse n'est pas un nombre entre 0 et 20.

Demander la saisie d'une note **jusqu'à ce que** la réponse soit un nombre entre 0 et 20.

Revenir au sommaire

Boucle tant que

Syntaxe première version :

```
Do While Condition
Instructions
[... Exit Do]
[Instructions]
Loop
```

La condition est ici testée **au début** c'est à dire à l'entrée de la boucle.

Avec `While` (tant que) la boucle est répétée **tant que** la condition est **vraie**.

Si la condition n'est pas vraie au départ les instructions de la boucle ne sont pas exécutées.

Exemple :

```
Do While MotProposé <> MotDePasse
MotProposé = InputBox("Donnez votre mot de passe")
Loop
```

Cela présuppose MotProposé initialisé par une valeur autre que MotDePasse

(par exemple la valeur par défaut "").

Syntaxe deuxième version :

```
Do
Instructions
[... Exit Do]
[Instructions]
Loop While Condition
```

La condition est alors testée **à la fin** de la boucle.

Avec `While` (tant que) la boucle est répétée **tant que** la condition est **vraie**.

Les instructions de la boucle sont donc exécutées au moins une fois.

Exemple :

```
Do
```

```
MotProposé = InputBox("Donnez votre mot de passe")
    Loop While MotProposé <> MotDePasse
```

Cet exemple ne présuppose aucune initialisation de MotProposé.

Revenir au sommaire

Boucle jusqu'à ce que

Syntaxe première version :

```
Do Until Condition
Instructions
[... Exit Do]
[Instructions]
Loop
```

La condition est ici testée **au début** c'est à dire à l'entrée de la boucle.

Avec `Until` (jusqu'à) la boucle est répétée **jusqu'à ce que** la condition soit **vraie**.

Si la condition est vraie au départ les instructions de la boucle ne sont pas exécutées.

Exemple :

```
    Do Until MotProposé = MotDePasse
MotProposé = InputBox("Donnez votre mot de passe")
Loop
```

Cela présuppose MotProposé initialisé par une valeur autre que MotDePasse

(par exemple la valeur par défaut "").

Syntaxe deuxième version :

```
Do
Instructions
[... Exit Do]
[Instructions]
Loop Until Condition
```

La condition est alors testée **à la fin** de la boucle.

Les instructions de la boucle sont donc exécutées au moins une fois.

Avec Until (jusqu'à) la boucle est répétée **jusqu'à ce que** la condition soit **vraie**.

Exemple :

```
Do
MotProposé = InputBox("Donnez votre mot de passe")
Loop Until MotProposé = MotDePasse
```

Cet exemple ne présuppose aucune initialisation de MotProposé.

Revenir au sommaire

Boucle For … Each … Next

C'est une extension de la boucle For ... To … Next.

Syntaxe :

```
For Each Elément In Ensemble
Instructions
[ ... Exit For]
[Instructions]
Next [Elément]
```

Ensemble est le plus souvent un tableau.

Exemple :

```
Dim TabHasard(100) As Integer
Dim Cellule As Integer
Dim Réponse As String
Randomize
For Each Cellule In TabHasard
                    Cellule = Rnd * 100 + 1
Next
For Each Cellule In TabHasard
```

```
Réponse = Réponse & Cellule & "   "
Next
MsgBox (Réponse)
```

Revenir au sommaire

Conclusion

Selon le problème à traiter vous aurez **le choix** entre ces différentes structures de contrôle.

Il s'agira de choisir sinon **la plus élégante** du moins celle qui ne provoquera pas de disfonctionnement de votre programme.

```
' Trouver ce qui ne va pas dans les exemples suivants :
```

Exemple1 :

```
Dim MotProposé, Réponse As String
     Réponse = "Titanic"
Do
               MotProposé = InputBox("Donnez votre réponse")
     Loop While MotProposé = Réponse
```

Exemple2 :

```
     Dim Note As Single
     Do Until Note >= 0 And Note <= 20
               Note = InputBox("Taper une note entre 0 et 20")
     Loop
```

Revenir au sommaire

webmaster : webmaster@vbasic.org

COURS Z11 N° 7

Traitement des chaînes de caractères

Sommaire

Nécessité de ces traitements

Les données manipulées par un programme sont essentiellement de type **numérique** ou **chaîne de caractères**.

Si les types numériques sont très utilisés par les programmes scientifiques, le type chaîne est incontournable pour des étudiants en Lettres et Sciences Humaines.

Une variable chaîne de caractères se déclare de type String.

Exemple 1 :

```
Dim MotProposé As String
```

La variable contient alors une chaîne de longueur **variable** selon l'affectation qui suivra.

Exemple 2 :

```
Dim Lettre As String * 1
```

La variable contient alors une chaîne de longueur 1 c'est à dire **un seul caractère**.

Exemple 3 :

```
Dim Adresse As String * 30
```

La variable contient alors une chaîne de **longueur 30**. Si l'on n'affecte que 18 caractères dans une telle chaîne, le reste est rempli d'espaces. Si l'on affecte plus de 30 caractères le surplus est tronqué.

Revenir au sommaire

Comparaison des chaînes de caractères

Longueur d'une chaîne :

La longueur d'une chaîne est donnée par la fonction **Len**.

Exemple :

```
Phrase = "Alea jacta est."
```

La fonction **`Len(Phrase)`** retournera la valeur **15**.

Il est évident que si deux chaînes de caractères n'ont pas la même longueur elles sont différentes. Par contre deux chaînes de même longueur ne sont pas forcément identiques.

Comparaison binaire :

On compare deux chaînes par la fonction **StrComp**.

Elle renvoie la valeur numérique **0** si les deux chaînes sont rigoureusement identiques et la valeur numérique **1** si les chaînes diffèrent même par un seul octet.

Exemple :

```
Phrase1 = "My tailor is rich."

Phrase2 = "My Tailor is rich."

Phrase3 = " My tailor is rich."
```

La fonction **`StrComp(Phrase1,Phrase2)`** retournera la valeur …

La fonction **`StrComp(Phrase1,Phrase3)`** retournera la valeur …

Revenir au sommaire

Recherche d'une chaîne de caractères

- La fonction InStr permet de rechercher si une chaîne de caractères **existe** à l'intérieur d'une autre chaîne de caractères. Cette fonction retourne la **position** de la première occurrence de la chaîne recherchée.

  Syntaxe :

  ```
  InStr(Chaîne1, Chaîne2)
  ```

  `Chaîne1` est la chaîne de caractères à traiter (sur laquelle porte la recherche).

Chaîne2 est la chaîne de caractères recherchée dans **Chaîne1**.

Exemple :

```
Chaîne1 = "Sed lex dura lex" ' Auteur inconnu

Chaîne2 = "lex"

Chaîne3 = "gex"
```

La fonction **InStr(Chaîne1,Chaîne2)** retournera la valeur "……."

La fonction **InStr(Chaîne1,Chaîne3)** retournera la valeur **0**.

- La fonction Ucase met tout le texte en majuscules et permet de rechercher indépendamment une lettre minuscule ou majuscule.
- La fonction Lcase met tout le texte en minuscules et permet…

Revenir au sommaire

Extraction d'une chaîne de caractères

- La fonction Right donne la partie **droite** d'une chaîne de caractères. Le nombre de caractères de cette partie doit être précisé.

  Exemple :

  ```
  Chaîne = "To be or not to be" 'Shakespeare
  ```

  La fonction **Right(Chaîne,5)** retournera la valeur "……."

- La fonction Left donne la partie **gauche** d'une chaîne de caractères. Le nombre de caractères de cette partie doit être précisé.

  Exemple :

  ```
  Chaîne = "To do is to be" 'Platon
  ```

  La fonction **Left(Chaîne,5)** retournera la valeur "……."

- La fonction Mid extrait une **partie** d'une chaîne de caractères.

  Le nombre de caractères de cette partie doit être précisé ainsi que la position du premier caractère extrait.

  Syntaxe :

  ```
  Mid(Texte, Position, Nombre)
  ```

Texte est la chaîne de caractère à traiter.

Position est la position dans la chaîne à partir de laquelle il faut extraire des caractères.

Nombre est le nombre de caractères à extraire.

Exemple :

```
Chaîne = "Do bee do bee do" 'Frank Sinatra
```

La fonction **Mid(Chaîne,4,2)** retournera la valeur "…."

La fonction **Mid(Chaîne,8,9)** retournera la valeur "……..."

Revenir au sommaire

Applications

La plupart des problèmes sur les chaînes de caractères se traitent à l'aide des fonctions ci-dessus. Leur maniement est assez délicat et les résultats quelquefois difficiles à prévoir. Seule la pratique de leur programmation vous permettra d'en vérifier l'efficacité.

Exemple1 :

Que fait ce programme :

```
Dim Vers, Mot As String

Dim Position As Integer

Vers = "Pour qui sont ces serpents qui sifflent sur vos têtes ?"

Mot = InputBox("Taper le mot à rechercher")

Position = InStr(Vers, Mot)

If Position = 0 Then

          MsgBox("Pas trouvé !")

Else

           MsgBox("Trouvé à la position " & Position)

End If
```

Exemple2 :

Ce programme compte le nombre d'espaces dans une phrase.

```
Dim Phrase, Caractère As String
Dim Compteur, Longueur, i As Integer
Phrase = InputBox("Tapez votre phrase")
Longueur = Len (Phrase)
For i = 1 To Longueur
          Caractère = Mid(Phrase, i, 1)
          If Caractère = " " Then Compteur = Compteur + 1
 Next i
MsgBox("Cette phrase contient " & Compteur & " espaces.")
```

Revenir au sommaire

COURS Z11 N° 8

Compléments de programmation

Sommaire

Fonctions de date et d'heure

- La fonction **Date** donne la date système.

  Pour aujourd'hui `Date` retournera la valeur "**08/04/98**"

- La fonction **Time** donne l'heure système.

  Pour maintenant `Time` retournera la valeur "**11:10:00**"

- La fonction **Day()** donne le numéro du jour dans le mois.

  Pour aujourd'hui `Day(Date)` retournera la valeur **8**

- La fonction **Month()** donne le numéro du mois dans l'année.

  Pour aujourd'hui `Month(Date)` retournera la valeur **4**

- La fonction **Year()** donne le numéro de l'année.

  Pour aujourd'hui `Year(Date)` retournera la valeur **1998**

- La fonction **WeekDay()** donne le numéro du jour dans la semaine sachant que le dimanche porte le numéro 1.

  Pour aujourd'hui `WeekDay(Date)` retournera la valeur **4**

Cette dernière fonction permet de programmer facilement un "**calendrier perpétuel**".

Revenir au sommaire

Procédures et fonctions

Procédures :

Une procédure est un ensemble d'instructions qui participent à une même tâche.

Elle débute par le mot réservé Sub et se termine par End Sub.

Exemple de procédure événementielle :

```
Private Sub cmdQuitter_Click

    End

End Sub
```

Si un bloc d'instructions doit être utilisé à plusieurs endroits (par exemple dans plusieurs procédures événementielles) il est préférable d'en faire une **procédure publique** qui sera utilisable dans toute la feuille.

Exemple de procédure publique :

```
Public Sub SaisieNote()

Do

Note = InputBox("Tapez une note")

Loop Until Note>=0 And Note<=20

End Sub
```

Pour utiliser cette procédure il suffira de l'appeler par son nom : `SaisieNote`

Fonctions :

Une fonction est aussi un ensemble d'instructions mais qui retourne **une valeur** contenue dans le nom même de la fonction. Cette valeur doit être affectée au nom de la fonction avant la fin du bloc d'instructions.

Elles débutent par le mot réservé Function et se terminent par End Function. Il faut aussi préciser le type de la valeur retournée.

Exemple de fonction :

```
Public Function Carré(x) As Single

    Carré = x * x

End Function
```

Par exemple `Carré(7)` retournera la valeur **49**.

Revenir au sommaire

Les zones de l'éditeur de code

La programmation (événementielle ou non événementielle) s'implante dans ce que l'on appelle le **code de feuille** (menu **Affichage** - **Code**).

Dans le code de feuille on distingue 3 parties :

1. La partie supérieure est la zone des **déclarations** (séparée du reste par un trait horizontal). On y place les options et les déclarations de variables publiques.

   Exemple :

   ```
   Option Explicit

   Public NomJoueur As String
   ```

2. La partie suivante est la zone des **procédures publiques** (chaque procédure est séparée des autres par un trait horizontal). On y place les procédures publiques utilisables par toutes les procédures de la feuille (événementielles ou pas).

   Exemple :

   ```
   Public Function Carré(x) As Single

   Carré = x * x

   End Function
   ```

3. Enfin la partie suivante est la zone des **procédures événementielles** (chaque procédure est séparée des autres par un trait horizontal).

   Exemple :

   ```
   Private Sub cmdCalculer_Click()

   .....

   End Sub
   ```

Revenir au sommaire

Les types de variables

On recommande fortement de **déclarer** les variables utilisées dans le programme.

- `Integer` : de **-32 768** à **32 767**
- `Long` : de **-2 147 483 648** à **2 147 483 647**
- `Single` : décimaux en simple précision : 39 chiffres significatifs
- `Double` : décimaux en double précision : plus de 300 chiffres significatifs !
- `String` : de **0** à **65 535** octets
- `Variant` : de type nombre ou texte selon l'affectation faite

Revenir au sommaire

Algorithmique

Quand on a un problème à résoudre par programmation on doit tout d'abord trouver une **stratégie** pour y parvenir.

Il faut bien sur que cette stratégie soit "**programmable**" dans le langage de programmation choisi. Il faut donc bien connaître les caractéristiques et les possibilités de ce langage.

Le plus souvent on écrit un **algorithme** en français (c'est la stratégie adoptée) que l'on pourra ensuite **coder** dans le langage de programmation choisi.

***Problème 1* :**

Déterminer la carte la plus forte sur un ensemble de 6 cartes posées à l'endroit sur la table.

La résolution de ce problème par un être humain ou par un programme informatique est complètement différente.

Algorithme en français

Prendre la 1ere carte. Noter sa hauteur dans une variable.

De la 2eme à la dernière :

    Prendre une carte.

    Si sa hauteur est supérieure à celle notée dans la variable elle devient la plus forte

Recommencer

***Problème 2* :**

Extraire tous les verbes (conjugués ou non) d'un fichier texte de plusieurs pages.

Cela semble assez difficile à programmer (cela l'est en effet) mais réalisable par un "honnête" programmeur (il "suffit" de construire un lexique de tous les verbes…).

Le problème ici est le temps de traitement.

Problème 3 :

Trier en ordre décroissant les nombre contenus dans un tableau de 100 entiers.

Différentes stratégies sont à notre disposition. La plus classique est celle du **tri à bulles**.

Il s'agit de comparer chaque nombre à son suivant et de mettre en premier le plus grand des deux. On recommence ainsi jusqu'à ce que plus aucune permutation ne soit effectuée. Alors le tableau est trié.

Algorithme en français

```
Début

Faire

Permuté ← 0

    Pour i = 1 à 99

        Si Case i < Case i+1 alors

            Auxiliaire ← Case i

            Case i ← Case i+1

            Case i+1 ← Auxiliaire

            Permuté ← 1

        Fin de Si

    Nouveau i

Recommencer tant que Permuté = 1

Fin
```

Revenir au sommaire