Haut du formulaire



Bas du formulaire

**Cours VB.NET**

****

**Table des matièresPlierDéplier**

* [III. Principe et structure des programmes](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII)
  + [III-A. Les 'Objets'](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-A)
    - [III-A-1. Dans la vie courante](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-A-1)
    - [III-A-2. Dans Visual Basic.net](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-A-2)
  + [III-B. Programmation évènementielle : le premier programme](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-B)
    - [III-B-1. Principes de la programmation VB](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-B-1)
    - [III-B-2. Exemple : le premier programme](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-B-2)
    - [III-B-3. En pratique, que fait le programmeur ?](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-B-3)
      * [III-B-3-a. Il dessine l'interface utilisateur](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-B-3-a)
      * [III-B-3-b. Il écrit le code correspondant aux événements](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-B-3-b)
  + [III-C. Les instructions, les procédures : les 'Sub', les 'Function'](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-C)
    - [III-C-1. Les instructions](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-C-1)
    - [III-C-2. Les procédures](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-C-2)
    - [III-C-3. Procédures liées aux évènements](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-C-3)
    - [III-C-4. Procédures non liées](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-C-4)
    - [III-C-5. Procédures 'Sub'](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-C-5)
    - [III-C-6. Procédures 'Function'](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-C-6)
    - [III-C-7. Module standard](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-C-7)
    - [III-C-8. Private Public](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-C-8)
    - [III-C-9. Remarques](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-C-9)
    - [III-C-10. Lexique anglais=>Français](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-C-10)
  + [III-D. Les modules](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-D)
    - [III-D-1. Qu'est ce qu'un module ?](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-D-1)
    - [III-D-2. Comment créer un module standard](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-D-2)
    - [III-D-3. Lexique anglais=>Français](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-D-3)
  + [III-E. Notion de programmation 'procédurale' et de programmation 'objet'](https://plasserre.developpez.com/cours/vb-net/?page=principe#LIII-E)

[précédent](https://plasserre.developpez.com/cours/vb-net/?page=introduction)[sommaire](https://plasserre.developpez.com/cours/vb-net/)[suivant](https://plasserre.developpez.com/cours/vb-net/?page=edi)

**III. Principe et structure des programmes**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

**III-A. Les 'Objets'**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Image non disponible

VB utilise la notion d''OBJETS'.

Pour bien comprendre ce qu'est un objet, nous allons prendre des exemples dans la vie courante puis nous passerons à des exemples dans Visual Basic.

Image non disponibleVoir la vidéo : [au format 'Flash'](ftp://ftp-developpez.com/plasserre/v1class.swf.html)> ou [au format 'Avi'](ftp://ftp-developpez.com/plasserre/v1class.avi) en Visual Basic 2005.

La vidéo (identique à celle du chapitre sur les Classes) contient:

1)Objets, Classes.

2)Références, espaces de noms (à visionner plus tard).

**III-A-1. Dans la vie courante**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Ma voiture est un **objet**, cet objet existe, on peut l'utiliser.

Image non disponible

Ma voiture fait partie de "Les voitures", du type, du genre "Les voitures". "Les voitures" c'est une classe (Class) qui a ses caractéristiques :   
"Les voitures" ont une couleur, un moteur.., elles roulent en transportant des passagers...

mais je ne peux pas utiliser "Les voitures", la Classe; pour me déplacer, il faut avoir un objet "voiture".

**Avec la Classe je vais créer des Objets.**

Pour fabriquer ma voiture, je prends les caractéristiques de la class "Les voitures" (c'est comme un moule, une usine) et je fabrique une voiture, je la nomme 'MaVoiture'.

Sélectionnez

Dim MaVoiture As New Lesvoitures

MaVoiture est maintenant un nouvel objet de **type** 'Les voitures'.

Image non disponible

Class --> Objet

Type 'Les voitures'--> Objet 'Mavoiture'

Un Objet est créé selon un 'modèle' qu'on appelle une Classe.

On dit aussi qu'il faut **instancier** un objet à partir de la Classe.

'Mavoiture' est une **instance** de la classe 'Les voitures'.

(On dit aussi une '**occurrence**' de la classe)

De manière générale, une classe est une représentation abstraite de quelque chose.   
Tandis qu'un objet est un exemple utilisable de ce que représente la classe.

Remarque:

Avec la même classe on peut instancier plusieurs Objets.

**Propriétés (Attributs):**

Prenons MaVoiture.Image non disponible

Elle a des propriétés : une marque, une couleur, une puissance..

Pour indiquer la couleur de ma voiture on utilise la notation :

Sélectionnez

MaVoiture.couleur

Syntaxe : Objet.Propriété (Il y a un point entre les 2 mots)

Pour modifier la couleur et avoir une voiture verte on écrit :

Sélectionnez

MaVoiture.couleur= "Vert"

Et la voiture devient verte !! Image non disponible

MaVoiture.Phares.Avant indique les phares avant de la voiture.

MaVoiture.Phares.Avant.Allumé indique l'état des phares (Allumé ou non)

Si je fais :

MaVoiture.Phares.Avant.Allumé=True (Vrai) cela allume les phares.

**Méthodes :**

MaVoiture fait des choses : elle roule par exemple.

Pour faire rouler la voiture j'appelle la méthode 'Roule'.

MaVoiture.Roule

Syntaxe : Objet.Méthode (Il y a un point entre les 2 mots)

Si c'est possible pour cette méthode je peux indiquer à quelle vitesse la voiture doit rouler :

MaVoiture.Roule(100) 'j'ai ajouté un paramètre.

Le paramètre est un renseignement envoyé avec la méthode.

Il est possible parfois d'indiquer en plus si la voiture doit rouler en marche avant ou en marche arrière.

MaVoiture.Roule(10, Arriere)

Il y a donc 2 manières d'appeler la méthode Roule : avec 1 ou 2 paramètres, on dit que la méthode est surchargée; chaque manière d'appeler la méthode s'appelle 'signature'.

Première signature: MaVoiture.Roule(100)

Seconde signature: MaVoiture.Roule(10, Arriere)

**Evènement:**

Des évènements peuvent survenir sur un objet.

MaVoiture\_démarre est un évènement, quand la voiture se met en route (si par exemple j'ai fait MaVoiture.Roule(10, Arriere)), cet évènement démarre se déclenche automatiquement.

**Interface et implémentation:**

Ce que je vois de l'objet, c'est son interface exemple: la méthode Roule fait partie de l'interface d'une voiture. Par contre ce qui fait rouler physiquement la voiture se nomme implémentation, c'est le moteur, ce n'est ni visible ni accessible.

Si je crée une voiture je vais faire l'implémentation, je vais fabriquer le moteur, mais si je suis simplement utilisateur de l'objet voiture, je vais me contenter d'utiliser l'interface.

**Visibilité:**

Quand un objet est créé, il est visible et utilisable, uniquement dans la zone où il a été défini.

**Relation entre Objets:**

Héritage:

Une Classe (un moule) peut hériter d'une autre classe (d'un autre moule).

La classe Voiture hérite de la classe Véhicule, cela veut dire qu'une voiture est un véhicule; la classe voiture aurait les propriétés de la classe Véhicule.

Contenant-contenu:

On peut créer une Classe qui contient des Objets, une classe qui se compose d'objets. on parle de composition.

L'objet Voiture contient 4 objets Roue.

On dit que l'objet **encapsule** le contenu.

Collections:

Les collections sont des groupes d'objets semblables qui peuvent être énumérés.

Un parc automobile contient X Voitures; chaque voiture a un numéro d'item:

ParcVoiture.item(1) pour la première Voiture

ParcVoiture.item(2) pour la seconde Voiture

**Espace de noms:**

Un espace de nom regroupe des objets qui appartiennent au même 'domaine'. Cela sert a différencier des objets qui ont même nom mais ne font pas partie du même domaine: Si je parle de 'Porte' ce terme n'a pas la même signification si je suis dans l'espace de nom 'Composant électronique'(on y trouve des portes logiques) ou l'espace de nom 'Maison'.

Tous cela c'est simpliste, mais voila, vous avez compris ce qu'est un objet !

**III-A-2. Dans Visual Basic.net**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Une application Windows se compose de fenêtres (nommées aussi formulaires) dans lesquelles se trouvent des contrôles (bouton, liste, texte…)

Exemple de fenêtre avec 2 boutons, une zone de texte (un label) et une icône:



Dans une application Windows, il y a aussi des lignes de code utilisant des variables pour faire des calculs.

**En Visual Basic.Net tout est objet :**

- les fenêtres (on dit les formulaires),

- les variables,

- les contrôles (les boutons, liste, image, case à cocher..)

...

Il faut un **moule** pour faire un objet. Le moule c'est une **Classe**.

Le moule va servir à créer un objet,on dit **une instance**.

On peut créer, **instancier** une multitude d'objets avec le même moule.

Pour créer, démouler un objet, on utilise les mots clé Dim et As New.

Sélectionnez

Dim objet As New Classe

New est un constructeur.

Exemple : créer une fenêtre (un formulaire) :

Je dessine une fenêtre FormDémarrage (c'est la Classe, le moule)

puis

Sélectionnez

Dim F As New FormDémarrage

Crée une fenêtre qui se nomme 'F' à partir du moule, du modèle (FormDémarrage) que j'ai dessiné.

Autre exemple :

Sélectionnez

Dim B As New Button

Créer un bouton nommé 'B' avec les attributs habituels des boutons (Class Button)

Troisième exemple:

Comment créer une variable nommée Mavariable pouvant contenir un entier (Integer)

Sélectionnez

Dim MaVariable As New Integer

Dim MaVariable As Integer 'est correct aussi

Ici, pour une variable, on remarque que New peut être omis

**Tout objet a des propriétés.**

On utilise la syntaxe : Objet.Propriété (Il y a un point entre les 2 mots)

Si F est une fenêtre, F.BackColor indique la couleur de fond de la fenêtre.

S'il y a un bouton, la couleur de fond du bouton sera :

Sélectionnez

Bouton.BackColor

Ou

Sélectionnez

F.Bouton.BackColor

Noter la syntaxe : La couleur du bouton qui est dans la fenêtre F

En fait une propriété c'est une sorte de variable.

Comment modifier cette propriété?

Sélectionnez

Bouton.BackColor=Color.Red

'modifie la couleur de fond du bouton

Image non disponible

Autre exemple:

La propriété Visible: si elle a la valeur True (Vraie) l'objet est visible; si elle est à False l'objet n'est pas visible.

Sélectionnez

Bouton.Visible=False

'fait disparaître le bouton

=Ici il y a un bouton invisible!! oui, oui!!

**Les objets ont des méthodes parfois.**

Une méthode agit sur l'objet ou fait agir l'objet.

Prenons un exemple simplifié.

Les Listes ( liste déroulante) ont des lignes (Items) et une méthode Clear qui permet de les vider.

Si je veux vider toutes les lignes d'une liste nommée Liste1, je fais:

Sélectionnez

Liste1.Items.Clear()

Image non disponibleLes propriétés et méthodes se nomment les membres d'un objet.

**Certains objets ont des évènements:**

Reprenons notre bouton. Quand l'utilisateur click dessus, l'évènement Bouton\_Click survient.

Ce sont les objets contrôles (bouton, case à cocher..)et les formulaires qui ont des évènements.

**Interface et implémentation:**

Ce que je vois de l'objet, c'est son interface (le nom des propriétés, méthodes..) exemple: la méthode Clear fait partie de l'interface d'une ListBox. Par contre le code qui effectue la méthode (celui qui efface physiquement toutes les lignes de la listBox),ce code se nomme implémentation, lui n'est ni visible ni accessible.

**Visibilité:**

Quand un objet est créé, il est visible et utilisable, uniquement dans la partie du programme où il a été défini.

Par exemple habituellement, je peux voir et modifier la couleur d'un bouton uniquement dans le code de la fenêtre ou il est situé.

Pour les variables on parle de portée: la variable peut être locale (Private) ou de portée générale ('Public') visible partout.

**Relation :**

Héritage:

Une Classe (un moule) peut hériter d'une autre classe (d'un autre moule).

La classe Button hérite de la classe Control, cela veut dire qu'un bouton est un contrôle.

Si je crée un bouton, il aura les caractéristiques de la classe Button mais aussi de la classe Control.

Contenant-contenu:

Une Classe peut contenir d'autres classes.

Je peux décider qu'un Objet Rectangle va contenir 4 Objets Point

Collections:

Les collections sont des groupes d'objets semblables qui peuvent être énumérés.

Une fenêtre Windows (on dit un 'formulaire' contient une collection nommée 'Controls' composées de tous les objets (boutons, List, texte) contenus dans la fenêtre:

Sélectionnez

maFenetre.Controls.item(1)

contient par exemple le premier bouton

Sélectionnez

maFenetre.Controls.item(2)

contient par exemple une list.

**En résumé :**

Une classe est un élément logiciel qui décrit un type de données abstrait. Un type de données abstrait est un ensemble d'objets définis par une liste d'opérations et les propriétés de ces opérations Une classe est un élément logiciel qui décrit les caractéristiques d'un ensemble d'objets. En programmation orientée une classe déclare des propriétés communes à un ensemble d'objets. La classe déclare des attributs représentant l'état des objets et des méthodes représentant leur comportement. Une classe représente donc une catégorie d'objets. Il apparaît aussi comme un moule ou une usine à partir de laquelle il est possible de créer des objets. On parle alors d'un objet en tant qu'instance d'une classe (création d'un objet ayant les propriétés de la classe). (www.techno-science.net)

En Visual Basic.net tout est objet.

Les Classes sont des types d'objet.

Pour créer (instancier) un objet à partir d'une Classe, il faut utiliser les mots clé Dim ..As New:

Sélectionnez

Dim Objet As New Class

Un objet a :

* Des propriétés.
* Des méthodes.
* Des évènements.

Attention, par abus de langage, on emploie parfois indifféremment les mots 'Classe' et 'Objet', mais il est préférable de ne pas confondre le modèle et l'objet lui-même.

**Espace de nom:**

Les objets sont regroupés dans des **bibliothèques d'objets**; dans des 'espaces de noms'.

Il faut parfois importer la bibliothèque avant d'utiliser l'objet:

Sélectionnez

'Importe l'espace de nom Systel.IO

Imports System.IO

'On peut maintenant utiliser l'objet 'File':

Fl = File.Exists("vessaggi.gif")

Si l'import n'a pas été fait, **System.IO.File.Exists()** est accepté aussi.

**Lexique anglais=>Français:**

New = Nouveau.

**III-B. Programmation évènementielle : le premier programme**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Nous allons comprendre la programmation évènementielle: Comment fonctionne Visual Basic:

* Ce que voit l'utilisateur.
* Ce qu'a fait le développeur pour arriver à ce résultat.

Voir la vidéo au format 'Flash': Image non disponibleou au format AVI Image non disponibleen Visual Basic 2005

**III-B-1. Principes de la programmation VB**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Le programmeur va dessiner l'interface utilisateur (fenêtre, bouton, liste..), il va ensuite uniquement écrire les actions à effectuer quand certains événements se produisent sur cette interface.

C'est Visual Basic qui va entièrement s'occuper de la gestion des événements.

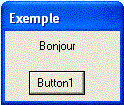
**III-B-2. Exemple : le premier programme**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Il affiche 'Bonjour' quand on clique sur un bouton.

Ce n'est pas original: le premier programme, dans tous les cours d'informatique, permet d'afficher 'Bonjour' (ou 'Hello Word').

**Que voit l'utilisateur du programme?**

L'utilisateur final, celui qui utilise le logiciel, voit une fenêtre avec un bouton, S'il appuie sur ce bouton il voit s'afficher " Bonjour".



**Que se passe t-il dans le programme?**

Quand l'utilisateur clique sur le bouton cela déclenche automatiquement un événement. (Button1\_Click), cet évènement contient du code qui affiche « Bonjour ».

**Que doit faire le programmeur pour arriver à ce résultat?**

Pour atteindre ce résultat, le programmeur va dessiner la fenêtre, le bouton, la zone d'affichage du texte (un label) puis il va simplement indiquer dans l'évènement Button\_Click d' afficher « Bonjour ».

Le fait de déterminer la procédure à appeler ou de réaliser l'appel est entièrement pris en charge par VB.

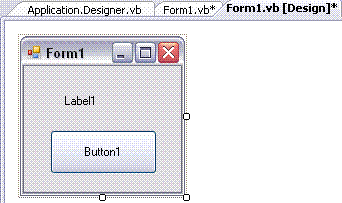
**III-B-3. En pratique, que fait le programmeur ?**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Image non disponibleVoir la vidéo : [au format 'Flash'](ftp://ftp-developpez.com/plasserre/v1bonjour.swf.html)> ou [au format 'Avi'](ftp://ftp-developpez.com/plasserre/v1bonjour.avi) en Visual Basic 2005.

Le programmeur est en mode 'conception' (ou mode Design): Il écrit le programme :

**III-B-3-a. Il dessine l'interface utilisateur** [**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

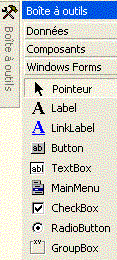
(Ce que verra l'utilisateur final, c'est l'interface utilisateur : une fenêtre avec des boutons, des listes, du texte..) :



Il ouvre un projet : une fenêtre 'Form1' apparaît.

Il ajoute un bouton :

Pour cela il utilise la Boite à outils:



Il clique sur 'Boite à Outils' à gauche, bouton Windows Forms, puis bouton 'Button', il clique dans Form1, déplace le curseur sans lâcher le bouton, puis lâche le bouton de la souris : le dessin d'un bouton apparaît.

Pour l'exemple,Il ajoute un label.

Un label est un contrôle qui permet d'afficher un texte.

Comme pour le bouton il clique sur 'Boite à Outils' à gauche, bouton Windows Forms, bouton 'Label' et met un contrôle label sur la fenêtre.

**III-B-3-b. Il écrit le code correspondant aux événements**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Il double-clique sur le bouton qu'il a dessiné :

Une fenêtre de conception de code s'ouvre et il apparaît :

Sélectionnez

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

End Sub

Cela correspond à la **procédure évènement** en rapport avec l'évènement 'On a cliqué sur le bouton1'.

Quand le programme fonctionne, quand l'utilisateur du logiciel clique sur le bouton1, le code situé entre Private Sub Button1Click et End Sub est effectué.

**Une procédure** est un ensemble de lignes de code qui commence par Sub et se termine par End Sub (ou Function..End Function).

Comment indiquer dans cette procédure d'afficher "Bonjour"?

Le label possède une propriété nommée '.text' qui contient le texte à afficher.

Il faut taper le code qui modifie cette propriété '.text' , qui y met la chaîne de caractère "Bonjour":

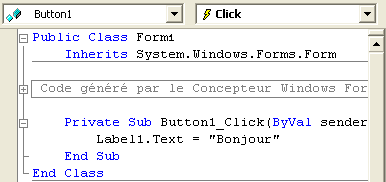
Sélectionnez

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click

Label1.Text = "Bonjour"

End Sub

Cela donne:



Voila votre premier programme est écrit.

**Comment exécuter ce programme?**

Il est possible de tester immédiatement le programme en mode débogage, sans quitter l'environnement de développement:

Utiliser le menu 'Déboguer' puis 'Démarrer' qui lance l'exécution du programme.

On peut aussi taper sur F5 pour lancer le programme.

Ou plus simplement cliquer sur la flèche:

Image non disponible

C'est plus rapide, lancer l'exécution avec le premier bouton, le second servant à arrêter temporairement l'exécution, le troisième à terminer l'exécution.

**En mode exécution :**

L'utilisateur voit bien une fenêtre avec un bouton, s'il clique dessus, « Bonjour » s'affiche.

Quand le programme est totalement écrit, terminé, testé, il est possible de le compiler et ainsi de créer un fichier exécutable (possédant une extension '.exe') qui fonctionne de manière autonome en dehors de l'environnement de développement.

C'est ce **fichier exécutable** qui est fourni à l'utilisateur.

Par opposition le code écrit par le programmeur, composé d'instructions Visual Basic, se nomme **le code source**.

**En résumé**

Le programmeur utilise des outils de dessin pour construire une interface utilisateur : des fenêtres avec des contrôles dessus: menus, boutons, case à cocher..

VB, pour chaque fenêtre ou pour chaque contrôle, génère une liste d'événements, (Evènement lié au chargement d'une fenêtre, évènement lié au fait de cliquer sur un bouton, évènement survenant quand on modifie un texte...)

Il suffit, dans la procédure événement qui nous intéresse, d'écrire le code qui doit être effectué lorsque cet événement survient.

Comme nous l'avons vu le code sert à agir sur l'interface (Afficher un texte, ouvrir une fenêtre, remplir une liste, un tableau), mais il peut aussi effectuer des calculs, évaluer des conditions et prendre des décisions, travailler en boucle de manière répétitive et ainsi effectuer les taches nécessaires .

**III-C. Les instructions, les procédures : les 'Sub', les 'Function'**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Qu'est ce qu'une instruction , une procédure?

Quelle différence entre les procédures liées aux évènements? non liées?:

Les 'Sub', les 'Functions'.

**III-C-1. Les instructions**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Une instruction est le texte tapé au clavier dans le 'code source' et permettant d'effectuer une opération, une déclaration, une définition.  
Elle contient des mots clés, des opérateurs, des variables, des constantes et des expressions des appels à des fonctions ou des méthodes. On verra cela en détails.

Sélectionnez

Dim A As Integer

est **une instruction** (de déclaration).

Sélectionnez

A = 1

est aussi une instruction qui effectue une opération.

C'est habituellement une **'ligne de code exécutable'.**.

Une instruction est exécutée lorsque le programme marche.

**Plusieurs instructions peuvent se suivre sur une même ligne**, séparées par ':'

Sélectionnez

Dim B As String : B="Bonjour"

**Si une ligne est très longue, on peut passer à la ligne grâce à ' \_'**

(caractère 'Espace' puis caractère"\_" puis immédiatement après, passage à la ligne) :

Sélectionnez

Dim B, C As String

B = "Bonjour monsieur ": C= \_

"le professeur"

est équivalent à:

Sélectionnez

Dim B, C As String

B = "Bonjour monsieur ": C= "le professeur"

En VB 2010, après certains mots il peut y avoir continuation de ligne implicite (plus besoin de \_ après la virgule, une parenthèse ouvrante, apres & ou { ou = ou +...).

Quand un programme tourne, les instructions sont effectuées ligne après ligne.

Sélectionnez

1 Dim B As String

2 B="Bonjour"

3 Dim A As Integer

4 A= 3

5 A= A + 1

La ligne 1 est exécutée puis la ligne 2 puis la 3, la 4...

Bien que l'on puisse avoir des numéros de ligne, ils ne sont plus utilisés actuellement et non visibles:

Sélectionnez

Dim B As String

B="Bonjour"

Dim A As Integer

A= 3

A= A + 1

Pour mettre **des commentaires** dans un programme, on le fait précéder de ' (on peut aussi utiliser le mot REM en début de ligne).

Sélectionnez

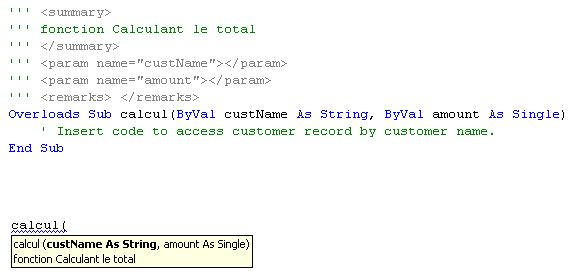
'Ceci est un commentaire,

'Cela n'est pas une instruction.

REM Ceci est aussi un commentaire.

Le commentaire ne sera pas exécuté.

Il peut aussi, à partir de VB 2005 , y avoir des commentaires en XML, ils sont dans ce cas précédés de ''' (3').

[](https://plasserre.developpez.com/cours/vb-net/images/vcomxml.jpg)

**III-C-2. Les procédures**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Une procédure est un ensemble d'instructions, de lignes de code, un groupement d'instructions bien définies effectuant une tache précise.

Les procédures sont bien délimitées:

**Il y en a de 2 sortes:**

**Les procédures Sub:**

Elles débutent par le mot Sub et se terminent par End Sub.

**Les procédures Function:**

Elles débutent par Function et se terminent par End Function.

Exemple:

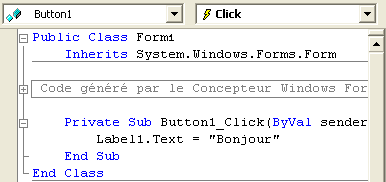
Sélectionnez

Sub Maprocédure

A=1

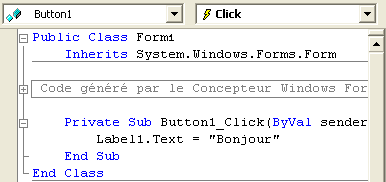
End Sub

Exemple concret d'une procédure: la procédure Button\_Click du premier programme.(Celui qui affiche 'Bonjour'; elle ne contient qu'une ligne de code. Le mot Sub est précédé de Private, on verra plus loin ce que cela signifie.



Vous avez vu que l'on peut dessiner l'interface, une fenêtre Form1 par exemple. En mode conception, après avoir dessiné l'interface, on doit avoir accès aux procédures.

Si on double-clique sur la fenêtre, on a accès aux procédures évènement liées à cette fenêtre, si on double-clique sur un objet (bouton, case à cocher…), on voit apparaître les procédures évènement de ce contrôle.



Quand on voit ces procédures, on peut y inclure du code.

Nous allons voir qu'il y a 2 types de procédures: les procédures liées aux évènements et celles qui ne sont pas liées.

**III-C-3. Procédures liées aux évènements**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Si on **double clique sur le fond d'une fenêtre (en vb 2010)**,(Celle du programme 'Bonjour') on voit apparaître **les procédures liées** à cette fenêtre et aux contrôles contenus dans cette fenêtre :

Sélectionnez

Public Class Form1

Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) \_

Handles MyBase.Load

Label1.Text = ""

End Sub

Private Sub Button1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) \_

Handles Button1.Click

Label1.Text = "Bonjour"

End Sub

End Class

Détaillons:

Sélectionnez

Public Class Form1

End Class

Ce n'est pas une procédure mais la 'Classe' définissant la fenêtre.

En VB 2003, il y a une 'région' que vous déroulez, en cliquant sur le petit +, vous pouvez y lire le code permettant de créer la fenêtre, les contrôles.. C'est généré automatiquement par VB. (Le chapitre VII-C sur les formulaires explique en détails le code généré par VB, mais c'est un peu complexe pour les débutants pour le moment!!)

En VB 2005 2008 et 2010 cette partie générée par VB n'est pas visible directement.

Il faut comprendre qu'à un formulaire (fenêtre) et aux contrôles qui sont dans ce formulaire correspond du code généré par VB. Ce code (sur lequel vous n'intervenez habituellement pas ) permet de créer le formulaire et les contrôles.

Chaque fenêtre a une procédure Form\_Load qui est exécutée lorsque la fenêtre est chargée, on y met généralement le code initialisant la feuille.

Sélectionnez

Private Sub Form1\_Load

End Sub

Il y a bien d'autres procédures liées à la fenêtre :

Dérouler la liste box en haut à gauche de la fenêtre de code, cliquer sur **(Form1 events),** si vous déroulez maintenant la liste à droite vous aurez **tous les événements qui génèrent une procédure** :

Load Lors du chargement de la fenêtre.

Unload Lors du déchargement de la fenêtre.

Activated Lorsque la fenêtre devient active.

GotFocus Lorsque la fenêtre prend le focus.

Resize Lorsque la fenêtre est redimensionnée.

Sélectionnez

Private Sub Button1\_Click

End Sub

C'est **la procédure liée au bouton** et qui contient le code à effectuer quand l'utilisateur clique sur le bouton.

C'est là que l'on écrit le code qui doit s'effectuer lorsque l'utilisateur clique sur le bouton.

De la même manière que pour la fenêtre, vous pouvez voir dans la liste en haut, tous les événements liés aux boutons qui génèrent une procédure :

Click Lorsque l'utilisateur clique sur le bouton.

DoubleClick Lorsque l'utilisateur double-clique sur le bouton.

MouseDown 'se déclenche si appui du bouton gauche de la souris.

MouseUp 'se déclenche si relâchement du bouton gauche de la souris.

On voit donc que le formulaire (la fenêtre) et tous les contrôles d'une application ont chacun des procédures pour chaque évènement qui peut survenir.

**III-C-4. Procédures non liées**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Parfois on a besoin de code qui fait une tâche particulière, qui est utilisé à plusieurs endroits et qui **n'est pas liée à un événement**.

On crée dans ce cas **une procédure indépendante des évènements**.

Le système des procédures permet aussi de découper un problème complexe en quelques fonctions moins complexes et indépendantes les unes des autres.

Un programme vb est donc composé de procédures dont l'exécution est déclenchée par des évènements (ouverture d'une fenêtre, click sur un bouton..), ces procédures en appellent d'autres qui en appellent d'autres..

Ces procédures sont en fait des **sous-programmes** : si une ligne appelle une procédure, le programme 'saute' au début de la procédure, il effectue le code de la procédure puis revient juste après la ligne qui avait appelé la procédure et continue les lignes suivantes.

Exemple : plusieurs fois dans le programme j'ai besoin de calculer la surface d'un cercle à partir de son rayon et de l'afficher sur un label.

Plutôt que de retaper dans chaque procédure le code, je peux créer une procédure 'Sub' nommée AfficheSurfaceCercle.

Il suffit ensuite si nécessaire d'appeler la procédure qui effectue le calcul et affiche le résultat puis revient effectuer le code situé après l'appel.

**Comment appeler une procédure? avec:**

Sélectionnez

Call NomdeProcedure()

ou par

Sélectionnez

NomdeProcedure()

Call est facultatif.

Noter les parenthèses après le nom de la procédure.

**III-C-5. Procédures 'Sub'**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

**Comment créer cette procédure Sub?**

Dans la fenêtre de code, tapez :

Sélectionnez

Sub AfficheSurfaceCercle

puis validez. Vous obtenez :

Sélectionnez

Sub AfficheSurfaceCercle()

End sub

Le code de la procédure est compris entre le Sub et le End Sub.

Pour que le calcul se fasse, il faut fournir, (transmettre de la procédure qui appelle à la procédure Sub) la valeur du rayon.

Pour indiquer que la Sub doit recevoir un **paramètre** (un **argument** en VB) ajouter entre les parenthèses :

Sélectionnez

Sub AfficheSurfaceCercle( Rayon as Single)

Cela signifie qu'il existe une procédure qui reçoit comme paramètre une variable de type Single (Réel simple précision) contenant le Rayon.

Ajouter le code :

Sélectionnez

Label.text =(3.14\*Rayon\*Rayon).ToString

**Que fait cette ligne ?**

Elle fait le calcul: '3.14\*Rayon\*Rayon' ('\*' signifie multiplier), on transforme le résultat en chaîne de caractères (grâce à '.ToString') que l'on met dans la propriété .text du label : Cela affiche le résultat. (On verra toute cette syntaxe en détail ultérieurement)

On obtient:

Sélectionnez

Sub AfficheSurfaceCercle( Rayon as Single)

Label.text =(3.14\*Rayon\*Rayon).ToString

End sub

**Comment appeler cette Sub?**

N'importe quelle procédure pourra appeler la Sub AfficheSurfaceCercle en envoyant la valeur du rayon afin d'afficher la surface du cercle dans un label.

Exemple d'appel pour un rayon de 12 :

Sélectionnez

AfficheSurfaceCercle(12)

Affiche dans le label: 452.16

**III-C-6. Procédures 'Function'**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Parfois on a besoin que la procédure retourne un résultat, un seul, qu'elle donne en retour un résultat à la procédure appelante. Dans ce cas on utilise une Fonction.

Exemple: je veux créer une fonction à qui je fournis un rayon et avoir en retour la surface d'un cercle.

**Comment créer cette Function?**

Tapez Function SurfaceCercle puis validez, ajouter (Rayon As Single)

Tapez Return 3.14\*Rayon\*Rayon

Ce que la fonction doit retourner est après Return (ce que la procédure doit renvoyer à la procédure appelante.)

On obtient la fonction complète:

Sélectionnez

Function SurfaceCercle( Rayon as Single)

Return 3.14\*Rayon\*Rayon

End Function

**Comment appeler cette Function?**

Dans la procédure qui appelle, il faut une variable pour récupérer la valeur retournée par la Fonction:

Sélectionnez

S= NomdelaFonction()

N'importe quelle procédure pourra appeler la fonction et obtenir le résultat dans la variable S par exemple pour un rayon de 12 :

Sélectionnez

Dim S As Single

S=SurfaceCercle(12)

On appelle la fonction SurfaceCercle en envoyant le paramètre '12', ce qui fait qu'à l'entrée de la fonction, Rayon=12, le calcul est effectué et le résultat du calcul (452.16) est retourné grâce à Return. S récupère ce résultat.

Après l'appel de cette fonction, S est égal à 452.16

Il est possible de spécifier le type retourné par la fonction:

Sélectionnez

Function SurfaceCercle( Rayon as Single) As Single

As Single en fin de ligne après () indique que la fonction retourne un Single (un nombre en simple précision). Il faut donc que la variable qui reçoit la valeur retournée (S dans notre exemple) soit aussi un Single.

Il existe une **autre manière de retourner le résultat d'une fonction**, reprenons l'exemple précédent, on peut écrire:

Sélectionnez

Function SurfaceCercle( Rayon as Single)

SurfaceCercle= 3.14\*Rayon\*Rayon

Exit Function

End Function

Ici on utilise le nom de la fonction pour retourner le résultat, avec un signe '='.

Utilisez plutôt la méthode Return.

Exit Function permet aussi de sortir de la fonction, cela a le même effet que Return sauf que Return peut être suivi d'un argument de retour (et pas Exit Function).

**III-C-7. Module standard**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

La Sub **AfficheSurfaceCercle** affiche le résultat dans le formulaire où elle est située.

Par contre la fonction **SurfaceCercle** est d'intérêt général, n'importe quelle procédure doit pouvoir l'appeler, de plus elle n'intervient pas sur les contrôles des formulaires et n'est donc pas liée aux formulaires.

On la placera donc dans un **module standard** qui est un module du programme qui ne contient que du code. (Pas d'interface utilisateur)

Pour créer un module standard Menu Projet>Ajouter un module.

Y mettre les procédures.

**III-C-8. Private Public**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Avant le mot Sub ou Function on peut ajouter :

Private indiquant que la procédure est accessible uniquement dans le module.

C'est donc une procédure privée.

Les procédures liées aux évènements d'une feuille sont privées par défaut.

Public indiquant que la procédure est accessible à partir de toute l'application.

S'il n'y a rien devant Sub la procédure est publique

Exemple:

Sélectionnez

Private Function SurfaceCercle( Rayon as Single)

Return 3.14\*Rayon\*Rayon

End Function

**III-C-9. Remarques**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Pour sortir d'une procédure Sub avant la fin, utiliser Exit Sub (Exit Function pour une fonction).

Quand vous appelez une procédure, il faut toujours mettre des parenthèses même s'il n'y a pas de paramètres.

Sélectionnez

FrmSplash.ShowDialog ()

Eventuellement on peut faire précéder l'appel du mot clé Call, mais ce n'est pas obligatoire.

Sélectionnez

Call FrmSplash.ShowDialog ()

Nommage:

Quand vous créez une procédure utilisez "la casse Pascal" pour créer les noms de routine:

la première lettre de chaque mot est une majuscule (C'est donc une convention).

Sélectionnez

Sub CalculTotal()

**III-C-10. Lexique anglais=>Français**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Call = Appel.

Return= Retour.

Private= Privé.

Show= spectacle, exposition.

To show= montrer.

**III-D. Les modules**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

**III-D-1. Qu'est ce qu'un module ?**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

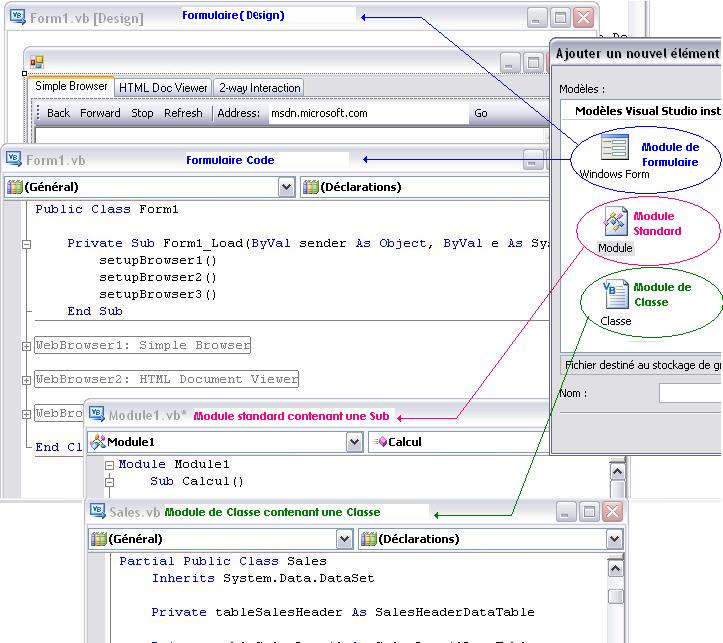
On a vu qu'un programme est décomposé en modules, chaque module contenant des procédures.

Chaque module correspond physiquement à un fichier '.vb'.

Il existe

* Les modules de formulaire.
* Les modules standards.
* Les modules de 'Classe'.

**Comment se présentent-ils?**

[](https://plasserre.developpez.com/cours/vb-net/images/vmodule1.jpg)

Un programme Visual Basic comporte donc :

* Les 'Modules de Formulaires':

contenant:

Le dessin des fenêtres de l'interface utilisateur (ou formulaire)contenant les contrôles (boutons, listes, zones de texte, cases à cocher…)

Le code qui comprend :

Les procédures liées aux événements de la feuille (Button\_Click..).

Les procédures indépendantes des événements. Ce sont des Sub() ou des Function().

Sélectionnez

Class Form1 'Nom du Formulaire

Inherits System.Windows.Forms

Public A as String

....

Private Button1\_Click 'Procédure liée à un évènement

..

End Sub

Sub MaRoutine 'Procédure indépendante

...

End Sub

End Class

* Les modules standards.

Ils servent de stockage de procédures. Procédures "d'intérêt général".

Ces procédures sont des Sub() ou des Function() qui peuvent être appelées à partir de n'importe quel endroit (Pourvu qu'elles soient 'Public').

Ils peuvent aussi servir à déclarer les objets ou déclarer les variables 'Public' qui seront utilisées donc accessibles par la totalité du programme.

Exemple:

Sélectionnez

Module Module1 'Nom du Module

Public A as String

....

Sub MaRoutine 'Procédure indépendante

....

End Sub

End Module

* les modules de Classe

Ils ont vocation à fabriquer des objets, on verra cela plus loin (Chapitre sur la programmation objet).

Exemple:

Sélectionnez

Class MaClasse 'Nom de la Classe

Public A as String

....

End Class

On remarque que les Class, formulaires, Modules, Sub, Functions sont délimités par:   
Une ligne de début comportant le type et le nom du module.   
Une ligne de fin contenant End et le Type.

Exemple:

Sélectionnez

Module Module1 'Nom du Module

...

End Module

Sub MaRoutine 'Procédure

....

End Sub

**III-D-2. Comment créer un module standard**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Faire Menu Projet puis Ajouter un module. Donner un nom au module. C'est Module1.vb par défaut.

Sélectionnez

Module Module1 'Nom du Module

...

End Module

On remarque que le module est bien enregistré dans un fichier '.vb'.

Un module standard ne contient que du code.

**Comment ajouter une Sub dans un module Standard?**

Taper Sub Calcul puis valider, cela donne:

Sélectionnez

Sub Calcul()

End Sub

**Remarque** Les Sub, Functions et Modules sont utilisés dans un type de programmation dit 'procédurale' où on découpe le code. Il existe un autre type de programmation dit 'Objet' ou on crée et on utilise des Objets; on verra cela plus tard.

**III-D-3. Lexique anglais=>Français**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Return = Retour.

**III-E. Notion de programmation 'procédurale' et de programmation 'objet'**[**▲**](https://plasserre.developpez.com/cours/vb-net/?page=principe)

Il y a deux manières de travailler en VB.NET:

* En programmation '**Procédurale**':

Chaque problème est décomposé en 'Fonctions'(Les Subs et Fonctions).

La programmation structurée découpe les problèmes en fonctions (Sub et Function). Ce découpage s'il est systématiquement employé aboutit à la programmation fonctionnelle qui consiste en un emboîtement de fonctions que l'on peut voir comme des « boîtes noires » que l'on peut imbriquer les unes dans les autres. Chaque fonction contient du code VB qui permet d'effectuer le travail dévolu à la fonction.

Ces fonctions sont stockées dans des modules standards (ou dans les modules de formulaire).

Dans une application en programmation 'procédurale' il y a habituellement:

Des modules de formulaires.

Des modules standard contenant des Sub et Function.

NB: j'utilisais, dans la précédente version du cours, le terme de programmation 'fonctionnelle' pour une programmation utilisant des Sub et Fonction; dans Wikipedia la programmation fonctionnelle c'est autre chose aussi je parle maintenant de programmation 'procédurale'..

* En programmation '**Objet**':

On verra cela plus tard: on crée ses propres objets dans des modules de Classe, on utilise les membres (Propriétés et méthodes) de ces objets pour programmer.

Dans une application en programmation 'Objet' il y a habituellement:

Des modules de formulaires.

Des modules de classe permettent de créer des Objets.

Grâce aux Classes (qui contiennent le code), on crée des objets.

Ensuite on utilise les propriétés et méthodes des objets.

De toute façon, dans les 2 cas, que se soit dans des Sub ou des Classes, on utilise du code Visual Basic.

La mode est à la programmation Objet!!

[précédent](https://plasserre.developpez.com/cours/vb-net/?page=introduction)[sommaire](https://plasserre.developpez.com/cours/vb-net/)[suivant](https://plasserre.developpez.com/cours/vb-net/?page=edi)

Vous avez aimé ce tutoriel ? Alors partagez-le en cliquant sur les boutons suivants : [Viadeo](http://www.viadeo.com/shareit/share/?url=http%3A%2F%2Fplasserre.developpez.com%2Fcours%2Fvb-net%2F&Title=Cours+VB.NET) [Twitter](http://twitter.com/timeline/home?status=http%3A%2F%2Fplasserre.developpez.com%2Fcours%2Fvb-net%2F) [Facebook](http://www.facebook.com/sharer.php?u=http%3A%2F%2Fplasserre.developpez.com%2Fcours%2Fvb-net%2F) [Share on Google+](https://plus.google.com/share?url=http%3A%2F%2Fplasserre.developpez.com%2Fcours%2Fvb-net%2F)  Partager

https://compteur.developpez.com/compter-82

Les sources présentées sur cette page sont libres de droits et vous pouvez les utiliser à votre convenance. Par contre, la page de présentation constitue une œuvre intellectuelle protégée par les droits d'auteur. Copyright © . Aucune reproduction, même partielle, ne peut être faite de ce site ni de l'ensemble de son contenu : textes, documents, images, etc. sans l'autorisation expresse de l'auteur. Sinon vous encourez selon la loi jusqu'à trois ans de prison et jusqu'à 300 000 € de dommages et intérêts.

https://compteur.developpez.com/spe-0https://compteur.developpez.com/spea-0

[**Contacter**](https://club.developpez.com/contacts/contribuer) **le responsable de la rubrique VB.NET**

* [Nous contacter](https://club.developpez.com/contacts/)
* [Participez](https://www.developpez.com/participez/)
* [Hébergement](https://www.developpez.com/hebergement/)
* [Informations légales](https://www.developpez.com/legal/)
* Partenaire : [Hébergement Web](https://www.planethoster.com/fr/)

**© 2000-2019 - www.developpez.com**

**Partenaire :** [**Hébergement Web**](https://www.planethoster.com/fr/)

https://www.developpez.com/template/kit/lightbox-ico-loading.png

https://www.developpez.com/template/kit/lightbox-btn-close.png