#### Compétence : Développer un programme informatique

**Elément de la compétence 1 : Maitriser les connaissances en Visual Basic**

**Objectifs pédagogiques 1 : Découvrir l’environnement Visual Basic**

**Leçon n°1 : Introduction sur le Visual Basic.**

1. **Je saurai:**

* Reconnaitre l’historique de langage de programmation Visual Basic;
* Expliquer le rôle de langage de programmation Visual Basic.

1. **J’observe et je réfléchi:**

|  |
| --- |
| Image1.1 le logo de logiciel Visual Basic. |

**Activite1 : Observation**

J’observe les images des logos de logiciel Visual Basic de différentes version.

Je découvre qu’il existe plusieurs versions de Visual Basic.

**Activite2 : Reconnaitre l’historique du langage de programmation Visual Basic**

**Je définis le langage de programmation Visual Basic (VB)**

**Visual Basic (VB)** est un langage de programmation événementielle de troisième génération ainsi qu'un environnement de développement intégré (**IDE** : Integrated development environment), créé par Microsoft pour son modèle de programmation COM (Component Object Model : qui est une technique de composants logiciels développée par Microsoft et DEC, depuis 1994). Visual Basic est directement dérivé du **BASIC** (**Beginner's All-purpose Symbolic Instruction Code**) et permet le développement rapide d'applications, la création d'interfaces utilisateur graphiques, l'accès aux bases de données en utilisant les technologies DAO (**Data Access Object),** ADO (**ActiveX Data Object**) et RDO, ainsi que la création de contrôles ou objets ActiveX.

**J’explique l’histoire de Visual Basic**

* **Le BASIC**

BASIC est un acronyme pour Beginner's All-purpose Symbolic Instruction Code. Le BASIC a été conçu en 1963 par John George Kemeny et Thomas Eugene Kurtz au Dartmouth College pour permettre aux étudiants qui ne travaillaient pas dans des filières scientifiques d'utiliser les ordinateurs.

Les huit principes de conception du BASIC étaient :

* Être facile d'utilisation pour les débutants (Beginner) ;
* Être un langage généraliste (All-purpose) ;
* Autoriser l'ajout de fonctionnalités pour les experts (tout en gardant le langage simple pour les débutants) ;
* Être interactif ;
* Fournir des messages d'erreur clairs et conviviaux ;
* Avoir un délai de réaction faible pour les petits programmes ;
* Ne pas nécessiter la compréhension du matériel de l'ordinateur ;
* Isoler l'utilisateur du système d'exploitation.
* **Le Visual Basic**

Du BASIC est né le Visual Basic (VB). Le VB est directement dérivé du BASIC et permet le développement rapide d'applications, la création d'interfaces utilisateur graphiques, l'accès aux bases de données, ainsi que la création de contrôles ou d'objets ActiveX. Avec ces possibilités je peux créer de petites chose comme:

Sub Main()

    MsgBox("Hello World !")

End Sub

Ce code me permet d’ouvrir un MsgBox dont le contenu est « Hello World ! ».

**J’énumère les versions de Visual Basic (à trouver)**

Les langages de script tels que Visual Basic for Applications et VBScript sont syntaxiquement proches de Visual Basic, mais s'utilisent et se comportent de façon sensiblement différente.

**Activité 3 : Dégager le rôle de langage de programmation Visual Basic.**

Le Visual Basic est utilisé pour :

* créer des applications personnalisées en utilisant des paramètres basiques et experts ;
* fournir une manière rapide et simple de créer des applications Windows .NET  Framewok ;
* offrir l’interopérabilité entre la sécurité et le langage aux applications crées.

Un programme en VB peut être développé en utilisant les composants fournis avec Visual Basic lui-même. Les programmes écrits en Visual Basic peuvent aussi utiliser l'[API Windows](https://fr.wikipedia.org/wiki/Windows_API), ceci nécessitant la déclaration dans le programme des fonctions externes.

**Je donne les avantages du langage VB**

**Le langage de programmation VB offre des avantages suivants:**

* Visual Basic n'est pas particulièrement compliqué pour le programmeur moyen;
* La syntaxe VB est généralement beaucoup plus facile que les autres langages de programmation;
* L'environnement de Visual Basic est facile à comprendre que celui du langage C ;
* Le VB a beaucoup de forums en ligne qui fournit des tutoriels et des réponses à des problèmes rencontrés;
* Le code VB est facile à écrire ;
* La connectivité de l’application VB à une base de données est assez élevée.

**Je donne les inconvénients du langage VB**

**Le langage de programmation VB offre quelques défauts à savoir :**

Le Visual Basic exige généralement une grande quantité de mémoire pour la mise en place initiale et le fonctionnement efficace après l'installation;

* Il est un outil de développement basé sur l'interface avec les aspects graphiques qui nécessitent une quantité importante d'espace;
* La déclaration et l’initialisation en VB sont compliqués par rapport à ceux du langage C surtout le tableau;
* Le Visual Basic n'est pas très utile pour les programmes qui utilisent beaucoup de temps de traitement, tels que les jeux.

1. **Je retiens l’essentiel.**

* **Le Visual Basic (VB)** est un langage de programmation événementielle de troisième génération ainsi qu'un environnement de développement intégré, créé par Microsoft pour son modèle de programmation COM.
* Le VB est directement dérivé du BASIC et permet le développement rapide d'applications, la création d'interfaces utilisateur graphiques, l'accès aux bases de données, ainsi que la création de contrôles ou d'objets ActiveX.
* Le Visual Basic a pour rôle de :
* créer des applications personnalisées en utilisant des paramètres basiques et experts ;
* fournir une manière rapide et simple de créer des applications Windows .NET  Framewok ;
* offrir l’interopérabilité entre la sécurité et le langage aux applications crées.
* **Les avantages du VB sont:**
* Visual Basic n'est pas particulièrement compliqué pour le programmeur moyen;
* La syntaxe VB est généralement beaucoup plus facile que les autres langages de programmation;
* L'environnement de Visual Basic est facile à comprendre que celui du langage C ;
* Le VB a beaucoup de forums en ligne qui fournit des tutoriels et des réponses à des problèmes rencontrés;
* Le code VB est facile à écrire ;
* La connectivité de l’application VB à une base de données est assez élevée.
* **Les inconvénients du langage VB sont :**
* Le Visual Basic exige généralement une grande quantité de mémoire pour la mise en place initiale et le fonctionnement efficace après l'installation;
* Il est un outil de développement basé sur l'interface avec les aspects graphiques qui nécessitent une quantité importante d'espace;
* La déclaration et l’initialisation en VB sont compliqués par rapport à ceux du langage C surtout le tableau;
* Le Visual Basic n'est pas très utile pour les programmes qui utilisent beaucoup de temps de traitement, tels que les jeux.

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
2. Définir le VB
3. Quel est le rôle de Visual Basic ?
4. Enumérer les avantages et inconvénients du langage VB
   1. **En utilisant mes connaissances**
5. En utilisant le langage VB, écrire un code permettant d’afficher le mot « Bonjour !  »
6. Qui a conçu le langage de programmation Basic ?
7. Dans quelle Année est conçu le Basic ?
   1. **Pour aller plus loin**

Donner autres quatre types de langage de programmation.

Donner au plus trois langages de programmation non visuels.

Donner au moins quatre langages de programmation visuels.

**Leçon n°2 : Description de l’interface VB**.

1. **Je saurai:**

* Lancer le VB 2010 express ;
* Décrire une fenêtre VB 2010 express ;
* Fermer et réafficher les éléments essentiels d’une fenêtre VB 2010 Express

1. **J’observe et je réfléchi:**

|  |
| --- |
|  |
| Image2.1 Fenêtre de Visual Basic 2010 Express |

**Activite1 : Observation**

J’observe une fenêtre de Visual Basic 2010 Express.

Je découvre qu’elle contient plusieurs parties dont les principales sont : barre de titre, les différents onglets, barre des outils standards, form, boite à outils, propriétés, explorateur de solutions.

**Activite2 : Lancer le VB 2010 express**

**Je donne les différentes méthodes utilisées pour lancer le VB 2010 express**

**1er méthode : (En général)**

Je fais un double clic sur son icône se trouvant dans la barre de tache ou au bureau.

Une fenêtre apparait, je clique sur « **nouveau** **projet** » ensuite sur « **application Windows Form** » enfin sur « **OK** ».

**2eme méthode : (Windows 7 et antérieurs »**

Je clique sur « **démarrer** » puis « **tous les programmes** » ensuite « **VB 2010 Express** »

Une fenêtre apparait, je clique sur « **nouveau** **projet** » ensuite sur « **application Windows Form** » enfin sur « **OK** ».

**3eme méthode : (Windows 10)**

Je clique sur « **démarrer** », dans la zone « **rechercher** » je tape VB enfin je clique sur « **Microsoft Visual Basic 2010 Express** » qui s’apparait.

Une fenêtre apparait, je clique sur « **nouveau** **projet** » ensuite sur « **application Windows Form** » enfin sur « **OK** ».

**4eme méthode : (Windows 8 et 8.1)**

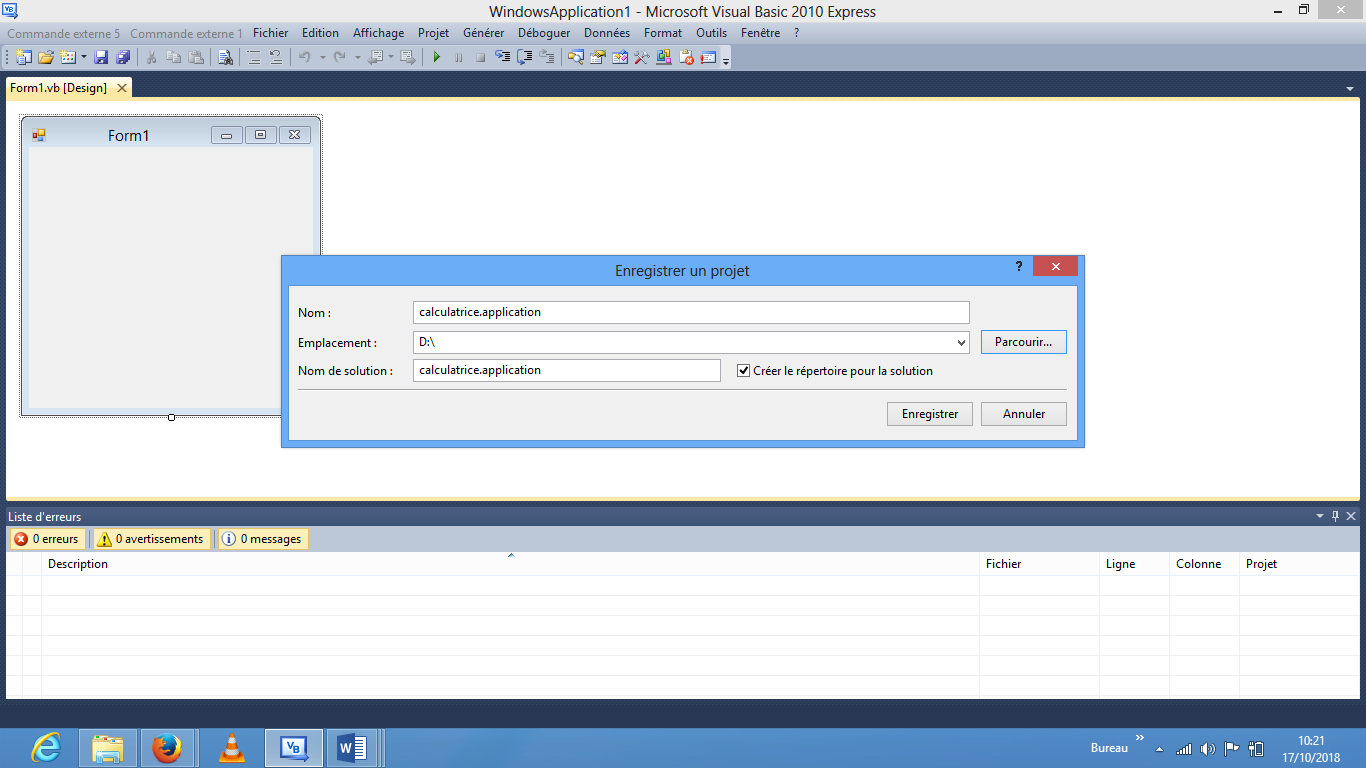
Je déplace le pointeur de la souris dans la partie supérieure à droite de mon écran puis je clique sur bouton « **rechercher** »,

Une fenêtre apparait, je clique sur « **nouveau** **projet** » ensuite sur « **application Windows Form** » enfin sur « **OK** ».

**Je montre comment enregistrer un projet**

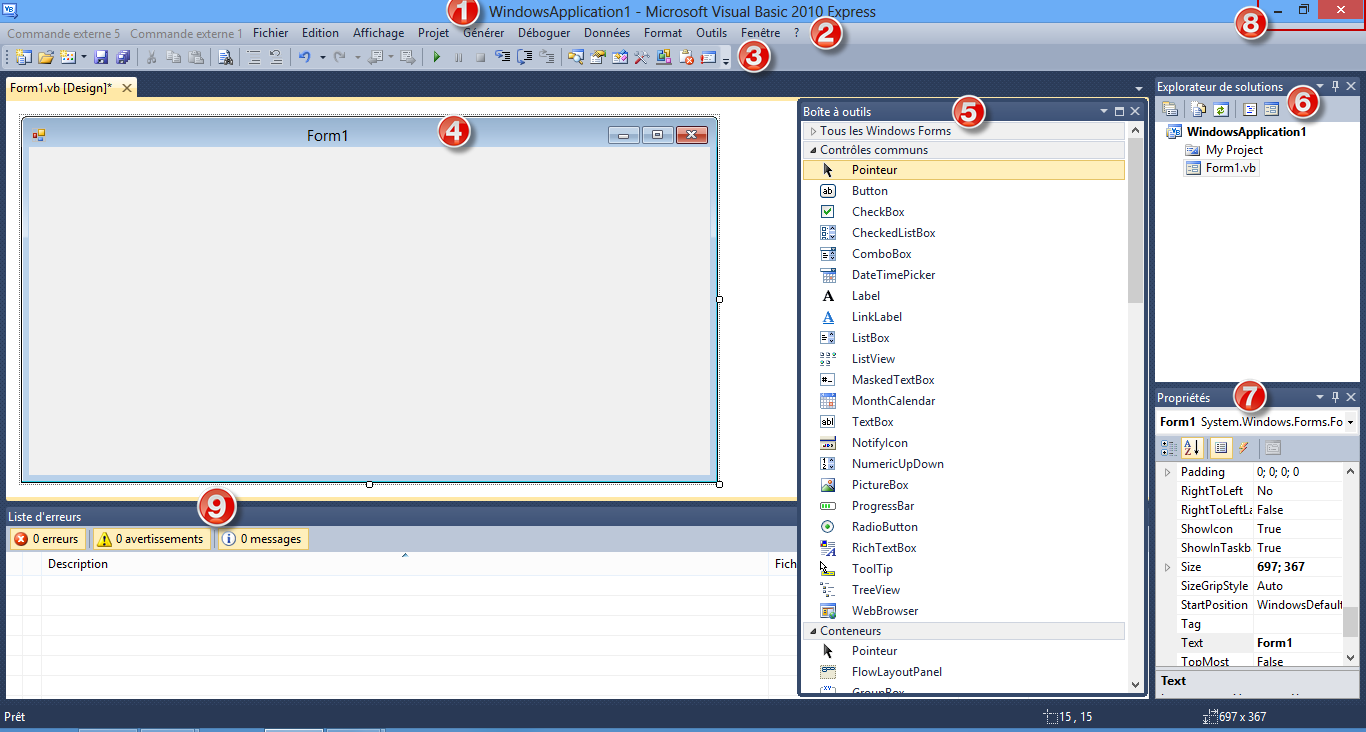
Apres avoir lancé le logiciel, on enregistre le projet en procédant les étapes suivantes :

* Clic sur « **Nouveau projet (Ctrl+N)** » ou bien « **Enregistrer tout (Ctrl+Maj+S)** » ;
* Taper le nom du projet (**Ex : Calculatrice.Application**) ;
* Choisir l’emplacement en cliquant sur parcourir (**soit disque local D, C, mes documents, Bureau)** ;
* Clic sur Enregistrer.



**Activité 3 : Décrire une fenêtre VB 2010 express**

**Je donne la description d’une fenêtre MS VB 2010 express**

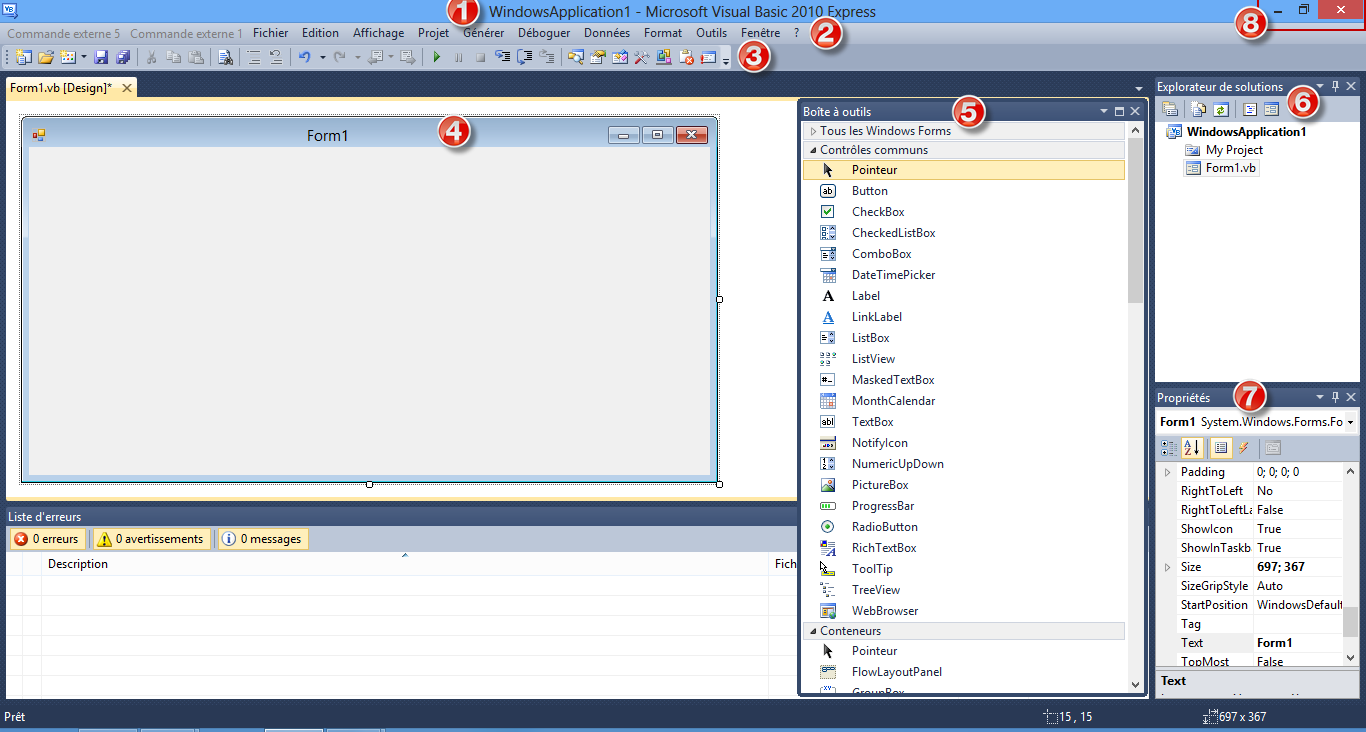


* + - 1. **La barre de titre**

La barre de titre est une zone supérieure de notre fenêtre dans laquelle est affiché le nom de l’application ainsi que le programme utilisé **EX** : WindowsApplication1-Microsoft Visual Basic 2010 Express.

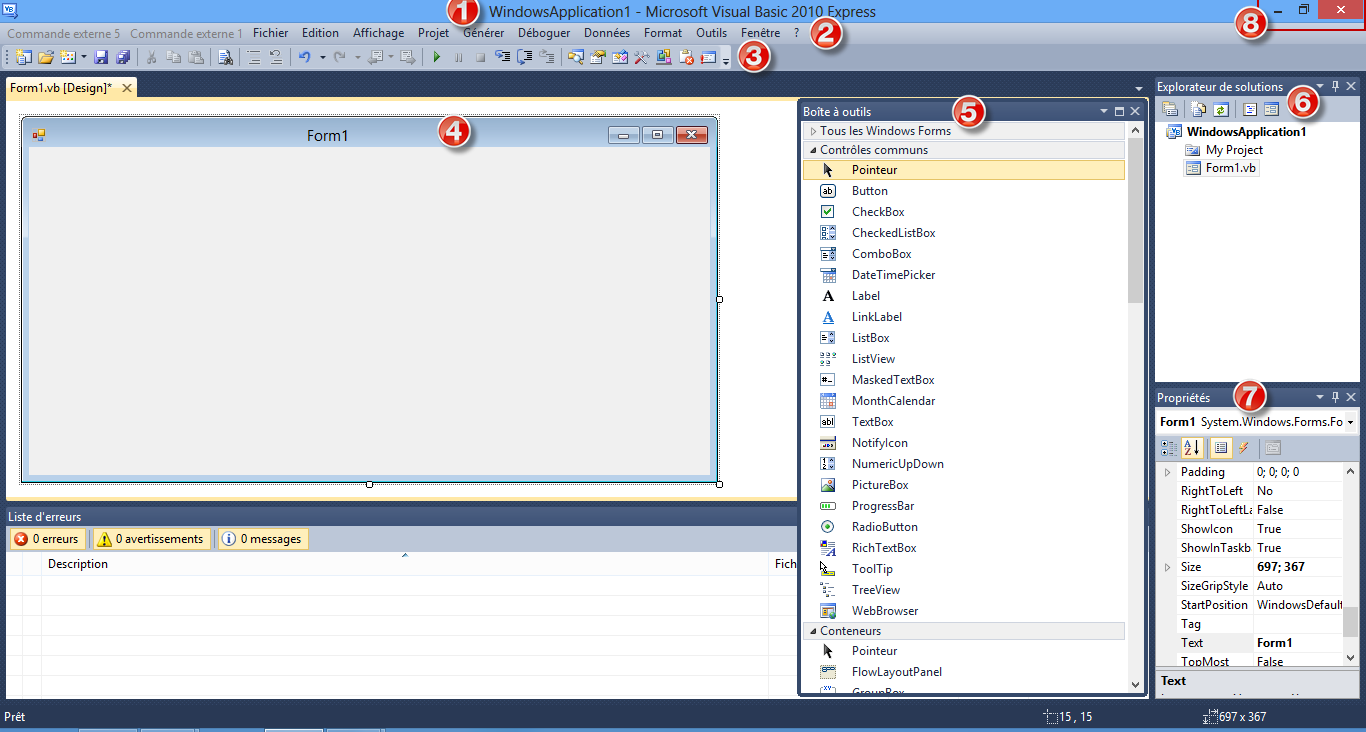
* + - 1. **Les différents onglets**

Sont des menus situent en dessous d’une da titre permettant d’accéder à divers sous menus, il s’agite de Fichier, Edition, Affichage, Projet, Générer, Déboguer, Données, Format, Outils, Fenêtre, Aide « ? ».



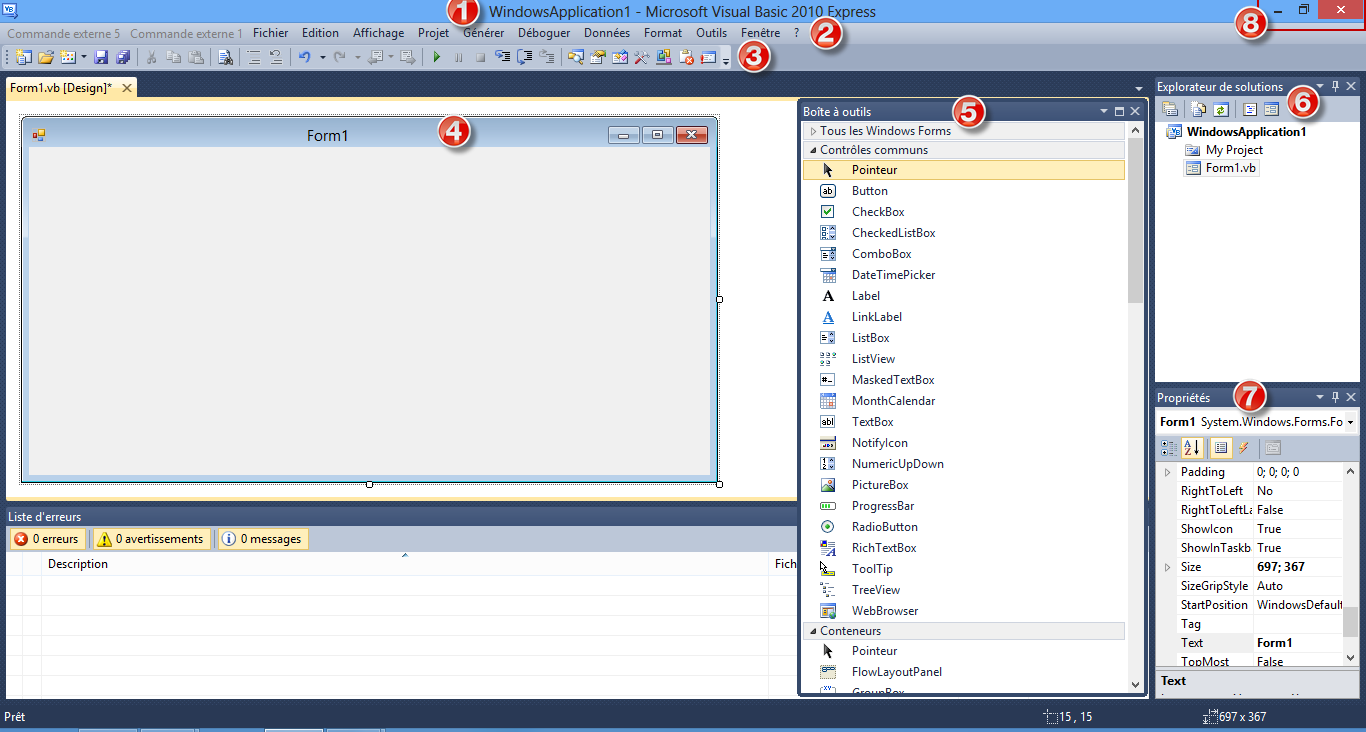
* + - 1. **La barre des outils standards**

Est une barre situe en dessous de la barre des onglets dans laquelle nous trouvons les outils d’accès rapide ; figure ci-dessous :



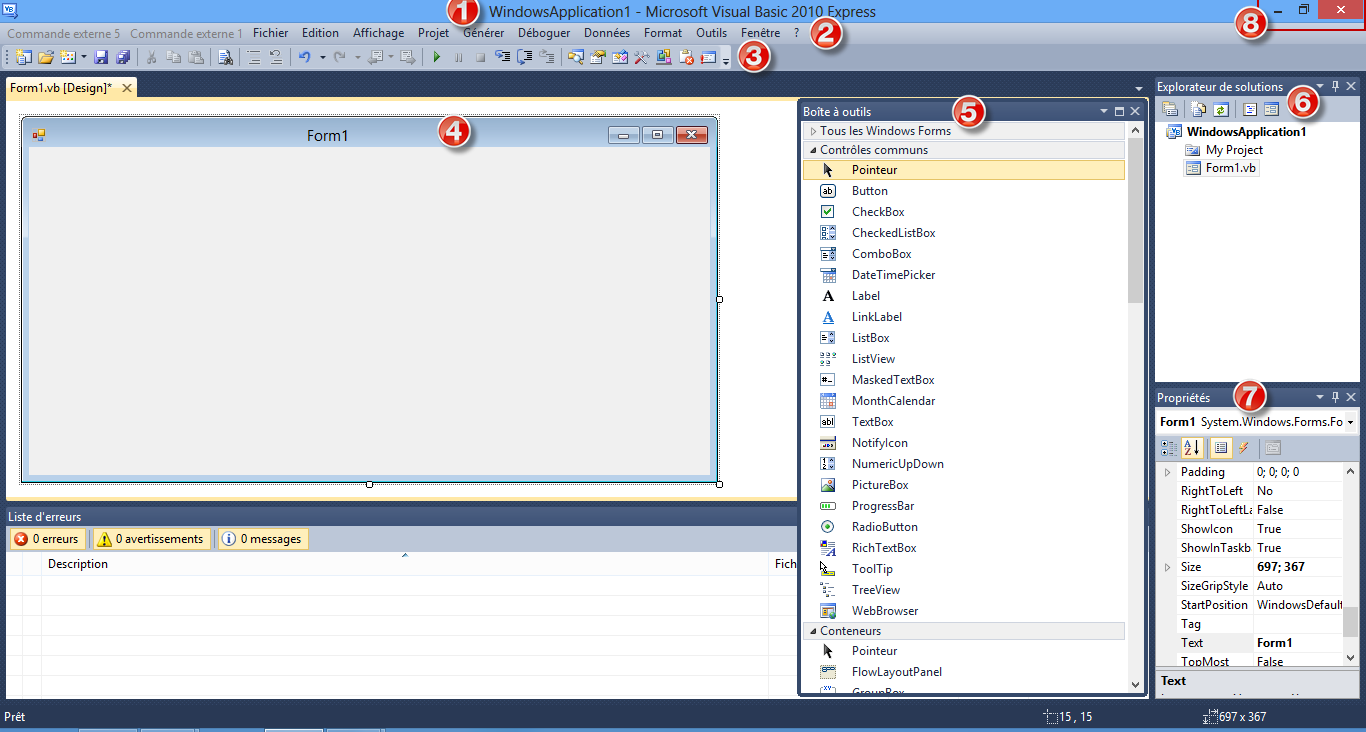
* + - 1. **Form**

Est un espace de travail dans lequel nous allons créer une fenêtre d’application.



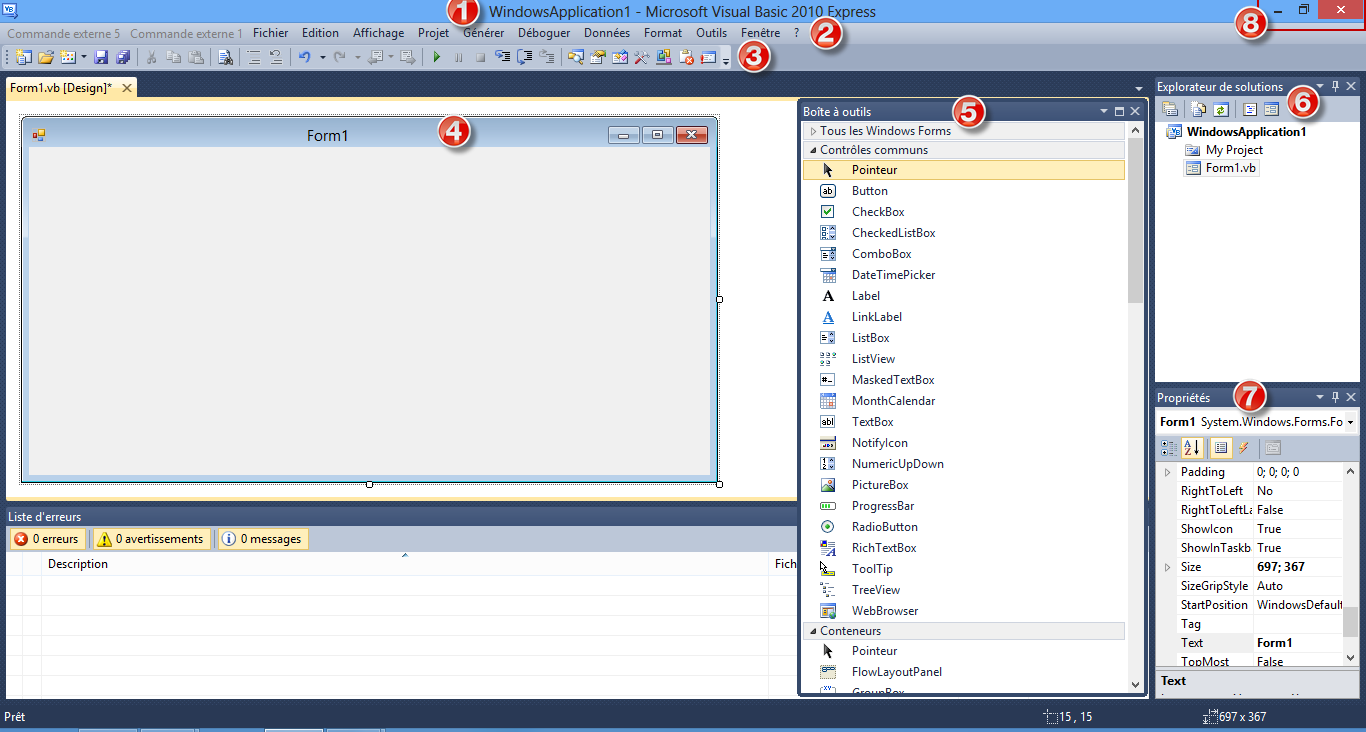
* + - 1. **La boite à outils**

Est une boite qui englobe les différents outils à insérer sur un form en vue de créer les interfaces graphiques de l’application.



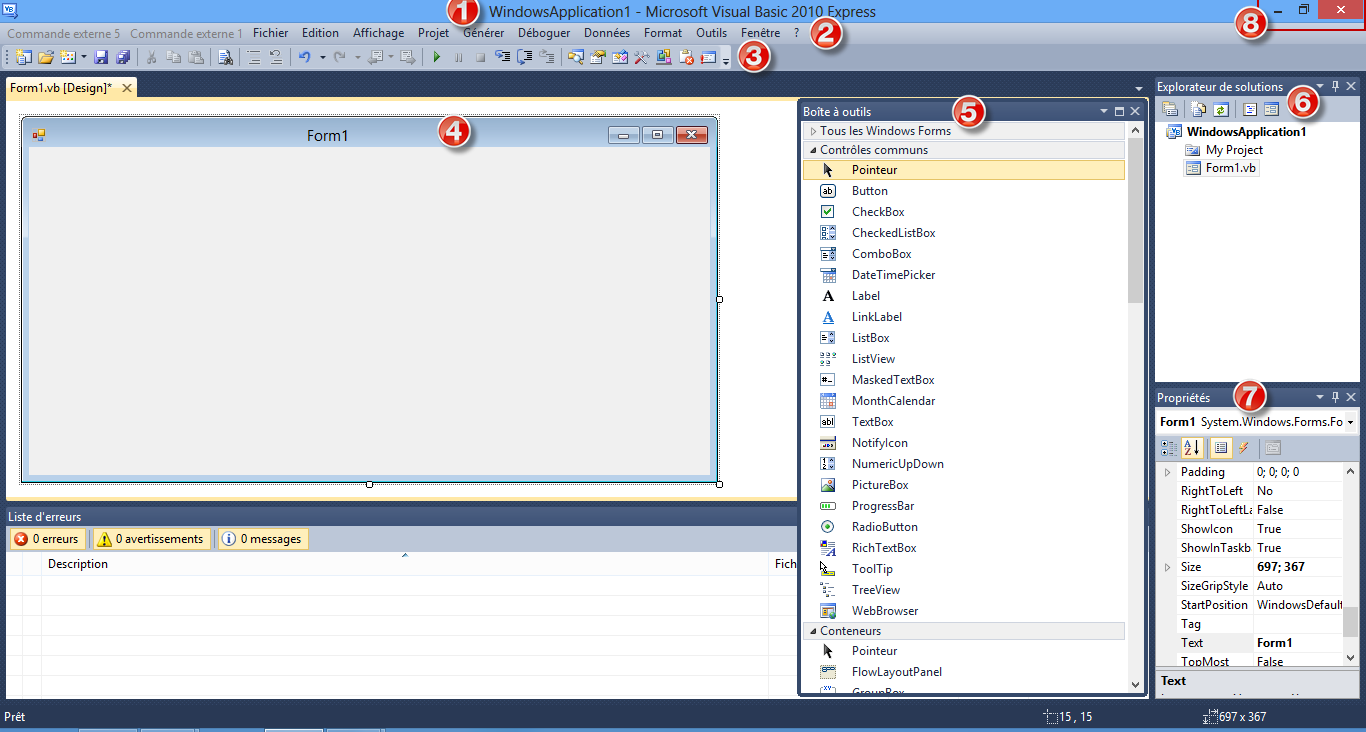
* + - 1. **Explorateur de solutions**

Est une petite fenêtre qui capture l’arborescence de notre projet, elle servira plus tard à gérer une base de données.



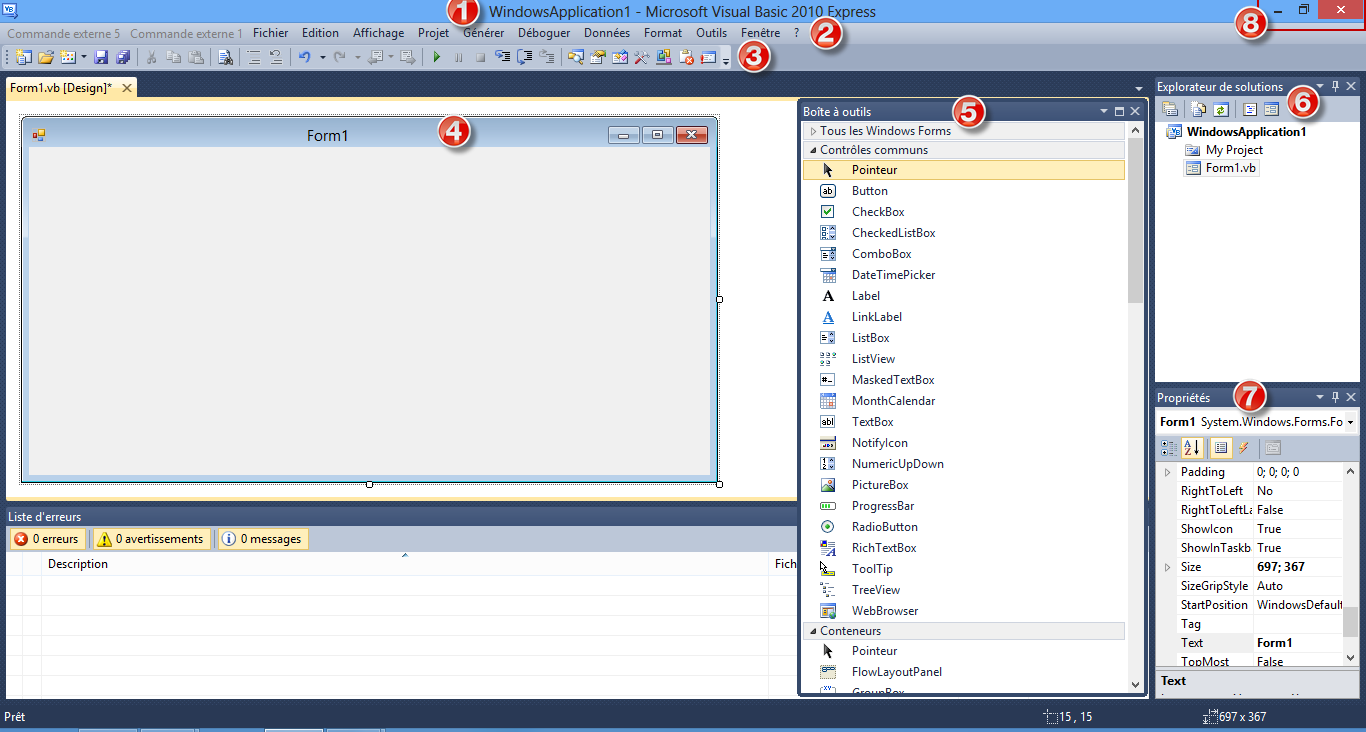
* + - 1. **Propriétés**

Est une partie essentielle permettant de faire la conception et de modifier les propriétés de nos objets ; Figure ci-dessous.



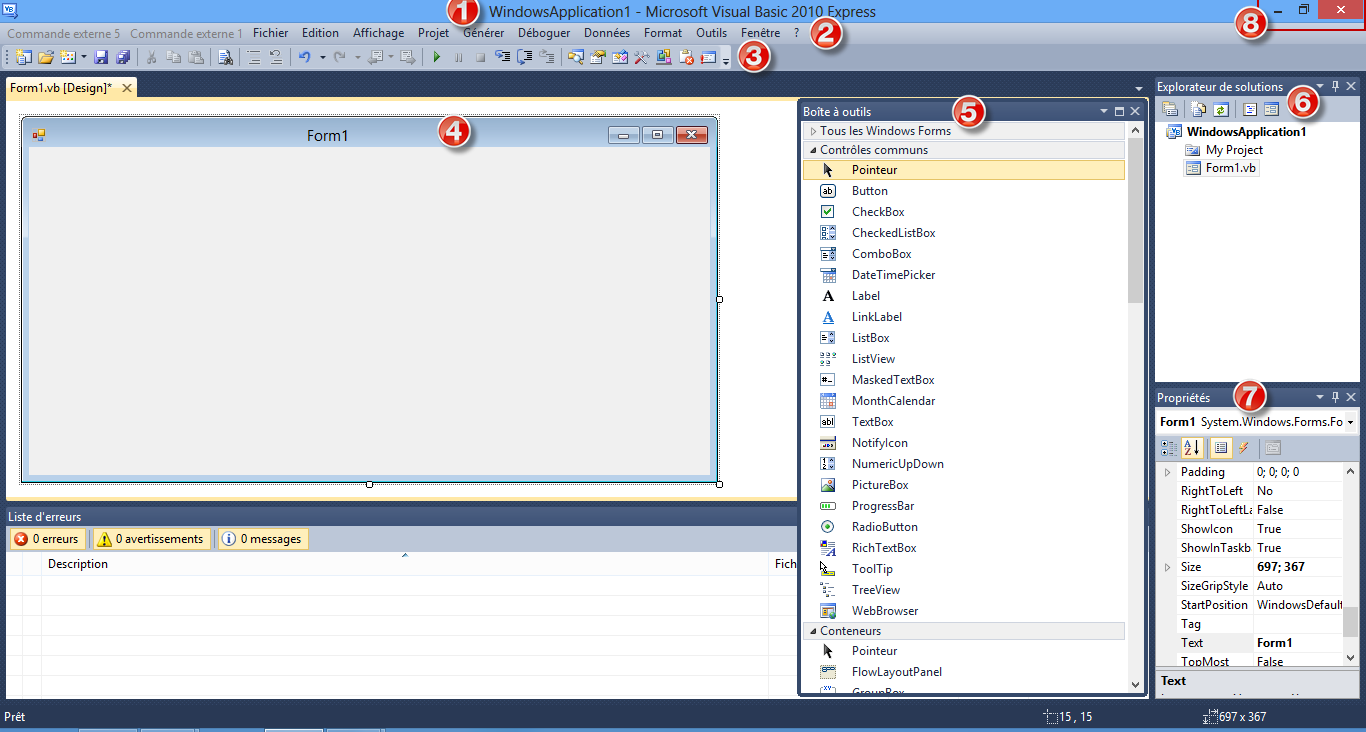
* + - 1. **Outil de contrôle**

Sont des boutons qui permettent le contrôle de notre projet tel que : « Fermer, Agrandir et réduire ».



* + - 1. **Liste des erreurs**

Est une petite fenêtre qui sert à afficher les erreurs rencontre dans les codes lors de l’exécution d’une application.



**Activité 4 : Fermer et réafficher les petites fenêtres essentiels d’une fenêtre VB 2010 Express**

J’explique comment fermer et réafficher les petites fenêtres essentiels d’une fenêtre VB 2010 Express en utilisant les étapes suivantes :

* Fermeture : pour fermer, il suffit de cliquer sur le bouton de chaque fenêtre
* Réaffichage : Pour réafficher, on clique sur l’onglet « Affichage » puis on clique sur :
* Explorateur de solutions
* Fenêtre de propriétés
* Liste d’erreurs
* Boite à outils
* Barre d’outils et cocher l’outil à afficher.

1. **Je retiens l’essentiel.**

* Les différentes méthodes utilisées pour lancer le VB 2010 express sont :

**1er méthode** : (En général)

Je fais un double clic sur son icône se trouvant dans la barre de tache ou au bureau.

Une fenêtre apparait, je clique sur « nouveau projet » ensuite sur « application Windows Form » enfin sur « OK ».

**2eme méthode** : (Windows 7 et antérieurs »

Je clique sur « démarrer » puis « tous les programmes » ensuite « VB 2010 Express »

Une fenêtre apparait, je clique sur « nouveau projet » ensuite sur « application Windows Form » enfin sur « OK ».

**3eme méthode** : (Windows 10)

Je clique sur « démarrer », dans la zone « rechercher » je tape VB enfin je clique sur « Microsoft Visual Basic 2010 Express » qui s’apparait.

Une fenêtre apparait, je clique sur « nouveau projet » ensuite sur « application Windows Form » enfin sur « OK ».

**4eme méthode** : (Windows 8 et 8.1)

Je déplace le pointeur de la souris dans la partie supérieure à droite de mon écran puis je clique sur bouton « rechercher »,

Une fenêtre apparait, je clique sur « nouveau projet » ensuite sur « application Windows Form » enfin sur « OK ».

* **Les étapes d’enregistrement d’un projet sont :**
* **Lancer le logiciel VB ;**
* Clic sur « **Nouveau projet (Ctrl+N)** » ou bien « **Enregistrer tout (Ctrl+Maj+S)** » ;
* Taper le nom du projet (**Ex : Calculatrice.Application**) ;
* Choisir l’emplacement en cliquant sur parcourir (**soit disque local D, C, mes documents, Bureau)** ;
* Clic sur Enregistrer.
* La fenetre du MS VB 2010 Express est décrite de façon suivante :

1. **La barre de titre :** La barre de titre est une zone supérieure de notre fenêtre dans laquelle est affiché le nom de l’application ainsi que le programme utilisé **EX** : WindowsApplication1-Microsoft Visual Basic 2010 Express.
2. **Les différents onglets :** Sont des menus situent en dessous d’une da titre permettant d’accéder à divers sous menus, il s’agite de Fichier, Edition, Affichage, Projet, Générer, Déboguer, Données, Format, Outils, Fenêtre, Aide « ? ».
3. **La barre des outils standards :** Est une barre situe en dessous de la barre des onglets dans laquelle nous trouvons les outils d’accès rapide ; figure ci-dessous
4. **Form :** Est un espace de travail dans lequel nous allons créer une fenêtre d’application
5. **La boite à outils :** Est une boite qui englobe les différents outils à insérer sur un form en vue de créer les interfaces graphiques de l’application.
6. **Explorateur de solutions:** Est une petite fenêtre qui capture l’arborescence de notre projet, elle servira plus tard à gérer une base de données.
7. **Propriétés :** Est une partie essentielle permettant de faire la conception et de modifier les propriétés de nos objets ;
8. **Outil de contrôle:** Sont des boutons qui permettent le contrôle de notre projet tel que : « Fermer, Agrandir et réduire ».
9. **Liste des erreurs:** Est une petite fenêtre qui sert à afficher les erreurs rencontre dans les codes lors de l’exécution d’une application.
10. **Je m’entraine et je m’évalue**

**4.1En restituant la leçon**

1. Comment lancer le VB 2010 express sous Windows XP professionnel
2. Enumérer les menus de M S Visual Basic 2010 Express

Rep1. Les méthodes peuvent être suivantes :

Faire un double clic sur son icône se trouvant dans la barre de tache ou au bureau.

Une fenêtre apparait, cliquer sur « nouveau projet » ensuite sur « application Windows Form » enfin sur « OK ».

Rep2. Les étapes sont :

* Lancer le logiciel VB ;
* Clic sur « **Nouveau projet (Ctrl+N)** » ou bien « **Enregistrer tout (Ctrl+Maj+S)** » ;
* Taper le nom du projet (**Ex : Calculatrice.Application**) ;
* Choisir l’emplacement en cliquant sur parcourir (**soit disque local D, C, mes documents, Bureau)** ;
* Clic sur Enregistrer.
* **Rep3. Ils sont :** Fichier, Edition, Affichage, Projet, Générer, Déboguer, Données, Format, Outils, Fenêtre, Aide « ? ».
  1. **En utilisant mes connaissances**

1. A quoi sert la boite à outils ?
2. Donner les étapes à suivre pour enregistrer un projet Visual Basic 2010 Express

Rep1. Elle sert à insérer sur un form les différents outils pour créer les interfaces graphiques de l’application.

Rep2. Les étapes sont :

* Clic sur « **Nouveau projet (Ctrl+N)** » ou bien « **Enregistrer tout (Ctrl+Maj+S)** » ;
* Taper le nom du projet (**Ex : Calculatrice.Application**) ;
* Choisir l’emplacement en cliquant sur parcourir (**soit disque local D, C, mes documents, Bureau)** ;
* Clic sur Enregistrer.
  1. **Pour aller plus loin**

Est-il possible d’enregistrer un projet sous un flash amovible ? Comment ?

Donner les différentes méthodes pour quitter l’application

Rep1. Oui, en suivant les étapes suivantes :

* **Brancher la cle USB,**
* Clic sur « **Nouveau projet (Ctrl+N)** » ou bien « **Enregistrer tout (Ctrl+Maj+S)** » ;
* Taper le nom du projet (**Ex : Calculatrice.Application**) ;
* Choisir l’emplacement en cliquant sur parcourir (**soit disque local D, C, mes documents, Bureau)** ;
* Clic sur Enregistrer
* COBOL(Common Busness Oriented Language),
* ALGOL (ALGorithmic Oriented Language),
* APL (A Programming Language),
* FACT(Fully Automated Compiling Technical),
* PL/1(Programming Language number 1).

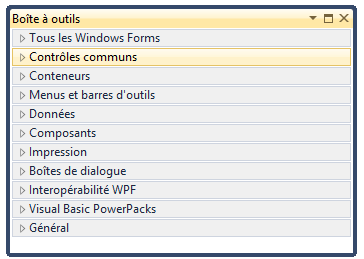
**Objectif pédagogique 2 : Décrire les éléments de l’application VB**

**Leçon n°3 : Etude d’une boite à outils**.

1. **Je saurai:**

* Différencier les outils d’une boite à outils ;
* Reconnaître le rôle de chacun outil ;
* Utiliser l’outil d’une boite à outils.

1. **J’observe et je réfléchi:**



|  |
| --- |
| Image2.1 Fenêtre d’une boite à outils sous MS Visual Basic 2010 Express |

**Activite1 : Observation**

J’observe une Fenêtre de boite à outils sous MS Visual Basic 2010 Express.

Je découvre qu’elle contient les différents groupes d’outils.

**Activite2 : Différencier les outils d’une boite à outils ;**

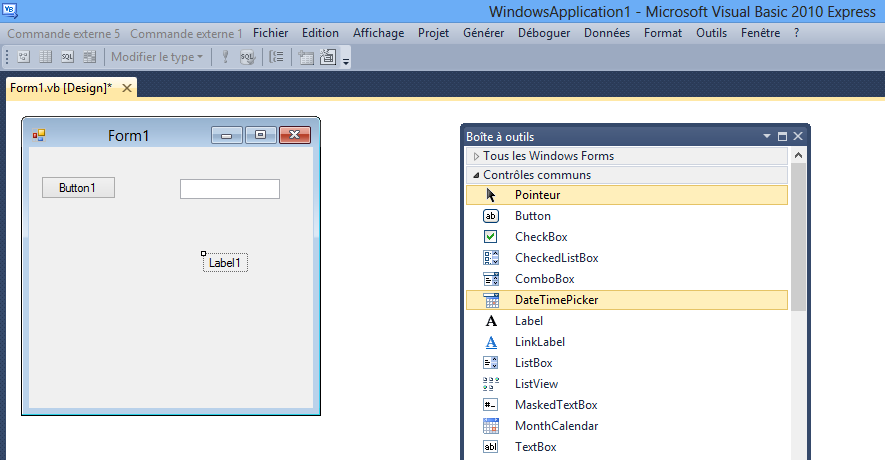
**Je différencie les outils d’une boite à outil sous MS Visual Basic 2010 Express.**

|  |  |
| --- | --- |
|  | **Bouton** permet de déclenche un évènement lorsque l’utilisateur clique dessus  **checkbox** permet à l’utilisateur de sélectionner ou de supprimer l’option associe  **checkedListBox** permet d’afficher une liste d’éléments avec une case à cocher placée à gauche de chaque élément.  **ComboBox** permet d’afficher une zone de texte modifiable avec une liste déroulante de valeurs automatisées.  **DateTimePicker** permet à l’utilisateur de sélectionner une date et une heure et d’afficher la date et l’heure dans un format spécifié  **Label** fournit des informations d’exécution ou un texte descriptif pour un contrôle.  LinkLabel affiche un contrôle label qui prend en charge la fonctionnalité du lien hypertexte, la mise en forme et le suivie.  **ListBox** affiche une liste à partir de sélectionner des éléments.  **ListView** affiche une correction des éléments dans les cinq modes d’affichage.  **MaskedTextBox** permet d’utiliser un masque pour distinguer entre une entrée d’utilisateur correcte et incorrecte  **MonthCalender** affiche un calendrier mensuel dans lequel l’utilisateur peut selectionner une date.  **TextBox** permet à l’utilisateur d’entrer du texte, de fournir des fonctions multilingue et de masquage des caractèeres du mot de passe.  **Notifycon** affiche une icône dans la zone de notification, sur la cote droite de la barre des tâches Windows pendant l’exécution.  **NumericUpDown** affiche une valeur numérique que l’utilisateur incrémenté et décrémenté en cliquant sur les boutons déroulants du contrôle.  **PictureBox** permet d’afficher une image.  **ProgressBar** affiche une barre de remplissage qui indique à l’utilisateur la progression d’une operation.  **RadioButton** permet à l’utilisateur de sélectionner une option unique à partir d’un groupe de choix en association avec d’autres cases d’option.  **Rich TextBox** fournit des fonctionnalités avancées d’entrée et d’édition de texte, comme la mise en forme des caractères et des paragraphes.  **ToolTip** affiche des informations lorsque l’utilisateur passe le pointeur sur un contrôle associée.  **TreeView** affiche pour l’utilisateur une collection hiérarchique d’éléments étiquetés qui contiennent éventuellement une image.  **WebBrowser** permet à l’utilisateur de parcourir les pages sein de votre formulaire.  **FlowLayoutPanel** Gère la disposition des composants et leur applique automatiquement une mise en page fluide.  **GroupeBox** affiche un frame autour d’un groupe de contrôles avec une légende de facultative.  **Panel** permet de grouper des collections de contrôles.  **SplitContainer** divise la zone d’affichage d’un conteneur en deux panneaux redimensionnables auxquels pouvez ajouter des contrôles.  **TabControl** gère et affiche à l’intention de l’utilisateur une collection d’onglets apparentés contenant des contrôles et des composants.  **TableLayoutPanel** manipule la disposition de ses composants les réorganise automatiquement sous forme de tableau.  **ContextMenuStrip** affiche un menu contextuel lorsque l’utilisateur clique avec le bouton droit sur le contrôle associé.  **MenuStrip** affiche des commandes et des options d’application groupées par fonctionnalité.  **PrintDialog** affiche une boite de dialogue qui permet à l’utilisateur de sélectionner une imprimante et de choisir d’autres options d’impression, telles que le nombre de copies et l’orientation de la page.  **OpenFileDialog** affiche une boite de dialogue qui demande à l’utilisateur d’ouvrir un fichier.  **SaveFicheDialog** affiche une boite de dialogue qui demande à l’utilisateur de sélectionner un emplacement pour l’enregistrement d’un fichier.  **Timer** est un composant qui déclenche un événement des intervalles définis par l’utilisateur  **BingingSource** permet d’encapsuler une source de donnés pour un formulaire et offre des fonctionnalités de navigation, de filtrage, de tri et de mise à jour. |

**Activité 3. Utiliser l’outil d’une boite à outils.**

**J’utilise les outils de la barre à outils.**

Pour insérer l’outil à une form1, je clique sur l’outil en besoin dans la boite à outils et je clique dans une form1.



**Le déplacement des contrôles:**

Plus de croix ni de petite flèche pour déplacer le contrôle ou ouvrir un menu. Pour déplacer le contrôle, appuyer bouton gauche dedans puis déplacer.

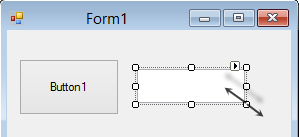
**L'alignement automatique des contrôles:**

Si on modifie la taille ou l'emplacement d'un contrôle, VB signale par un trait bleu que le contrôle modifié et le contrôle voisin sont alignés.



**Le redimensionnement d’un contrôle :**

Clique sur le contrôle  à redimensionner pour le sélectionner, clique sur l’un des point qui l’entourent une flèche à double sens s’afficher, je maintien le bouton gauche de la souris puis je glisse vers l’extérieur pour agrandir le contrôle ou vers intérieur dans le cas contraire.



1. **Je retiens l’essentiel.**

* **Le rôle des outils les plus utilisées en VB**
* **Bouton** permet de déclenche un évènement lorsque l’utilisateur clique dessus
* **checkbox** permet à l’utilisateur de sélectionner ou de supprimer l’option associe
* **ComboBox** permet d’afficher une zone de texte modifiable avec une liste déroulante de valeurs automatisées.
* **Label** fournit des informations d’exécution ou un texte descriptif pour un contrôle.
* **ListBox** affiche une liste à partir de sélectionner des éléments.
* **TextBox** permet à l’utilisateur d’entrer du texte, de fournir des fonctions multilingue et de masquage des caractèeres du mot de passe.
* **RadioButton** permet à l’utilisateur de sélectionner une option unique à partir d’un groupe de choix en association avec d’autres cases d’option.
* **La méthode utilisée pour insérer un outil (un contrôle)**

Pour insérer l’outil à une form, je clique sur l’outil en besoin dans la boite à outils et je clique dans une form.

1. **Je m’entraine et je m’évalue**

**4.1En restituant la leçon**

1. Donnez le role des outils suivant :
2. **ProgressBar**
3. **OpenFileDialog**
4. **Timer**
5. Comment insérer un contrôle dans un form ?

**Rep1 :**

**ProgressBar** affiche une barre de remplissage qui indique à l’utilisateur la progression d’une operation.

**OpenFileDialog** affiche une boite de dialogue qui demande à l’utilisateur d’ouvrir un fichier.

**Timer** est un composant qui déclenche un événement des intervalles définis par l’utilisateur

Rep2 pour insérer un contrôle dans un form je procède comme suis :

Je clique sur l’outil en besoin dans la boite à outils et je clique dans une form1.

* 1. **En utilisant mes connaissances**
     + 1. Donner les opérations qu’on peut faire sur un contrôle.

Rep1.

* **Le déplacement,**
* **L'alignement,**
* **Le redimensionnement,**
* **La suppression**
  + - 1. Comment procédez-vous pour déplacer un contrôle dans un form ?

Rep2. Pour déplacer le contrôle, appuyer bouton gauche dedans puis déplacer.

* 1. **Pour aller plus loin**

Comment accéder dans le code d’un contrôle ?

Rep : Pour accéder dans le code d’un contrôle, il suffit de double-cliquer sur ce contrôle.

**Leçon n°4: Découvrir les fenêtres de propriétés**.

1. **Je saurai:**

* Définir une fenêtre des propriétés ;
* Décrire les propriétés essentiels d’un contrôle sélectionné sous MS Visual Basic 2010 express ;

1. **J’observe et je réfléchi:**

|  |  |
| --- | --- |
|  |  |
| Image2.1 Fenêtre des propriétés sous MS Visual Basic 2010 Express | | | |

**Activité 1 : Observation**

J’observe une fenêtre des propriétés sous MS Visual Basic 2010 Express.

Je découvre que cette fenêtre contient plusieurs propriétés des objets sélectionnés

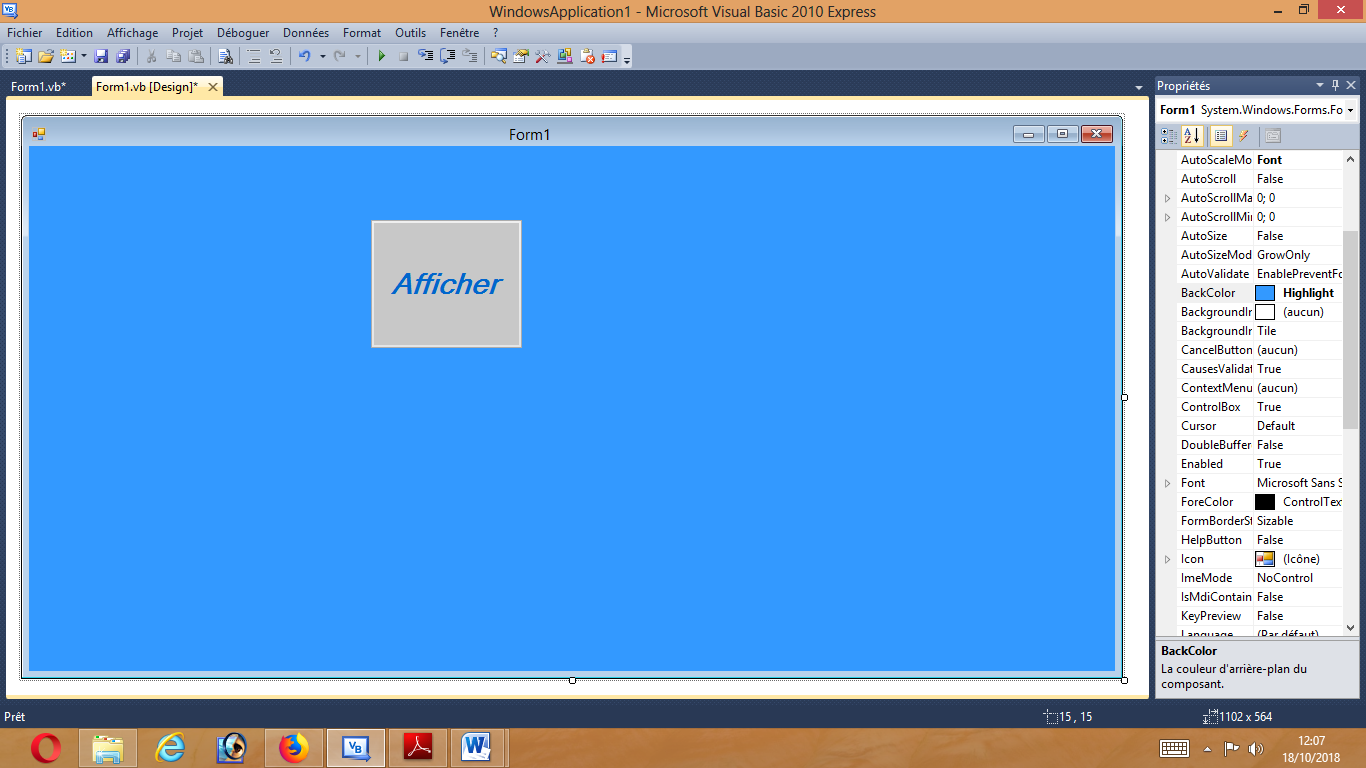
**Activité 2 : Définir une fenêtre des propriétés**

Une fenêtre des propriétés  est fenêtre contenant les noms des objets sélectionnés et qui affiche leurs propriétés.

**Activité 3 : Différencier les propriétés;**

**Je décris les éléments essentiels qui composent la fenêtre de propriétés sous MS Visual Basic 2010 express :**

* L’élément « **Name** » sert la nomenclature d’un objet sélectionné dans lequel se trouve dans un form «  ex : Button1 » ;
* L’élément « **Text** » sert à renommer l’objet sélectionné « ex : Button1 », par exemple « Afficher » ;
* L’élément « Dock » permet de positionner l’objet sélectionné soit au centre, en bas, à gauche, à droite dans un Form ;
* L’élément  « **image** » permet d’importer l’image ou photo se trouvant soit au Bureau, mes documents, disque local C, D, E,…. ;
* L’élément « **Font**» joue un même rôle que « police », il permet de faire la mise en forme du texte d’un objet sélectionné se trouvant dans un form ,soit en gras italique, en gras, oblique, normal , la taille, la police(ARGERIAN ,Candara,arial,……) ;
* L’élément « **Location** » permet de définir les coordonnées d’un objet sélectionné pour le déplacer dans un Form, soit en haut à droite, en bas, au centre, n’importe où selon les coordonnées ;
* L’élément «**Locked**» évite toute sorte de modification d’un objet sélectionné se trouvant dans un Form ;
* L’élément « **MaximumSize** » permet d’agrandir ou de diminuer l’objet sélectionné se trouvant dans un Form ;
* L’élément «**visible** » rend visible ou invisible un objet sélectionné se trouvant dans un Form ;
* L’élément «**BackColor** » permet d’attribuer la couleur de fond à un objet sélectionné ;
* L’élément «**foreColor** » permet de modifier la couleur de police des textes de l’objet sélectionné.



**Activité 4 : Expliquer comment utiliser les éléments d’une fenêtre de propriétés**

Je donne les étapes à suivre pour utiliser les élémentsd’une fenêtre

Je sélectionné l’objet à modifier ;

Je clique sur l’élément qui contient les propriétés voulus puis je sélectionne une propriété de mon choix.

1. **Je retiens l’essentiel**

|  |
| --- |
| * Les étapes à suivre pour utiliser les élémentsd’une fenêtre de propriétés sont * Sélectionner l’objet à modifier ; * Cliquer sur l’élément qui contient les propriétés voulus * Sélectionner une propriété au choix. * Une fenêtre des propriétés  est une fenêtre contenant les noms des objets sélectionnés et qui affiche leurs propriétés. |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**

Définir une fenêtre de propriétés

* 1. **En utilisant mes connaissances**

Donner le rôle des éléments des propriétés suivants :

1. Text ;
2. Font ;
3. BackColor ;
4. foreColor.

**Réponse :**

1. L’élément « **Text** » sert à renommer l’objet sélectionné « ex : Button1 », par exemple « Afficher »
2. L’élément « **Font**» joue un même rôle que « police », il permet de faire la mise en forme du texte d’un objet sélectionné se trouvant dans un form ,soit en gras italique, en gras, oblique, normal , la taille, la police(ARGERIAN ,Candara,arial,……)
3. L’élément «**BackColor** » permet d’attribuer la couleur de fond à un objet
4. L’élément «**foreColor** » permet de modifier la couleur de police des textes de l’objet
   1. **Pour aller plus loin**

Comment importer une image ou photo se trouvant soit au Bureau ou dans mes documents dans un form ?

**Réponse** : Pour importer une image ou photo se trouvant soit au Bureau ou dans mes documents dans un form on utilise l’élément de Propriétés «image ».

**Leçon n°5 : Distinction et déclaration de variables en VB**.

1. **Je saurai:**

* Définir une variable ;
* Différencier les variables utilisés en VB ;
* Déclarer une variable en VB.

1. **J’observe et je réfléchi:**

|  |
| --- |
| **Image2.1** **Fenêtre montrant l’exemple de déclaration des variables en VB**. |

**Activite1 : Observation**

J’observe un exemple de déclaration des variables en MS Visual Basic 2010 Express.

Je découvre qu’une variable est caractérisée par son nom et son type.

**Activité 2 : Définir une variable.**

Je défini une variable comme est une entité dans laquelle on stocke les données nécessaires au déroulement du programme.

Le nom d’une variable est caractérisé comme suis :

* Les noms de variables n’ont pas de longueur maximale,
* Ils doivent commencer par un caractère,
* Ils ne doivent pas comporter d’espace,
* Ils ne sont pas sensibles à la casse (Toto et toto sont la même variable).

**Activité 3 : Différencier les variables utilisés en VB**

Je différencie les variables utilisés en VB.

Comme en mathématique, la différence en informatique est qu’une variable ne contient qu'une valeur. Chaque variable a un Nom (identifiant) et un Type. Notons que les variables se différencient selon leurs types.

**Le type d’une variable indique:**

-La nature de l'information (un chiffre? du texte?)

- Les valeurs que peut prendre la variable (un entier, un réel..).

- Les opérations possibles (addition, concaténation)

**Exemple**: le Type 'Entier' (**Integer** en VB) peut contenir une valeur entière, positive ou négative, les opérations possibles sont l'addition, la soustraction, la multiplication.

**Les types disponibles en VB sont:**

**Type numérique**: 'Entier', 'réel' (**Integer**, Single en VB) Exemple d'un entier: 123

**Type alphanumérique**: 'Caractère' (Char en VB) contient 1 caractère Exemple d'un caractère: 'a' (avec des guillemets) 'Chaîne de caractères'(String en VB), contient plusieurs caractères. Exemple: 'toto' (avec des guillemets)

**Booléen** : (Boolean en VB) ne peut contenir que 'Vrai' ou 'Faux'

**Objet**. (Object en VB)

**Monétaire** (Décimal en VB)

**Date** (Date en VB)

**Matrice**, nombre imaginaire (depuis VB 2010)

A partir des types précédents on peut créer des types complexes (ou structurés):

-**Les Tableaux** (Array) qui contiennent plusieurs éléments.

-**Les Collections** qui contiennent plusieurs éléments aussi.

Exemple: la variable nommée 'Total' contient un réel dans un programme de comptabilité.

on remarque qu'il ne faut pas confondre 1 qui est une valeur numérique (sans guillemets) et "1" qui est le caractère '1' (avec des guillemets).

**Activité 3. Déclarer une variable en VB.**

Je déclare une variable en VB

Pour déclarer une variable en VB on choisit d’abord l’emplacement de la déclaration soit globale ou locale en suite on suit la syntaxe qui suis :

**Dim** NomVariable **As** Type

Avec : **Dim** mot clé du langage expliquant la dimension des données d’une variable.

NomVariable à remplacer par le nom à attribuer une variable ;

**As** mot clé du langage précisant le type d’une variable ;

Type à remplacer par le type de donnée à attribuer une variable.

L’existence d’une variable peut se dérouler sur trois niveaux :

* **Niveau Procédure :** cela veut dire que la variable est locale. Dès que l’on quitte la procédure en question, la variable disparaît, et son contenu avec elle. Pour déclarer une variable au niveau procédure, on tape au sein de la procédure considérée :

**Dim** NomVariable **As** Type

* **Niveau Form :** la variable est disponible pour toutes les procédures de la Form , mais pas pour les procédures se situant sur une autre Form. Pour déclarer une variable au niveau Form, on tape tout en haut de la Form, à l’extérieur des procédures :

**Dim** NomVariable **As** Type

* **Niveau Projet :** la variable est disponible, et sa valeur est conservée pour toutes les procédures de l’application, quel que soit leur emplacement. Pour déclarer une variable globale, il faut d’abord créer un module. Un module est un type de feuille destiné uniquement à recevoir du code, et qui n’a donc pas, contrairement aux Form, d’apparence graphique. C’est dans un module qu’on écrit la procédure principale, lorsqu’on en veut une, et qui de là pilote les différentes Form, elles-mêmes donnant accès aux procédures liés aux objets qu’elles contiennent. Sur ce module on écrit :

**Public** NomVariable **As** Type

Il faut donc pour chaque variable se demander à quel niveau on en a besoin, et faire les déclarations en fonction.

**Déclaration des variables indicées**

On appelle aussi cela des tableaux ! Ce peut être des tableaux de nombres, de chaînes, de booléens, bref, de tout ce qu’on veut. Quand on crée un tableau, soit on sait d’avance combien d’éléments il va englober, soit on veut qu’il soit dynamique (mais cela se paye bien sûr par une perte de rapidité à l’exécution). Tout tableau doit obligatoirement être déclaré, quel que soit par ailleurs le réglage de l’option de déclaration des variables.

Pour créer un tableau de 12 entiers, on écrira :

**Dim** MonTableau(11) **As** Integer

Pour créer un tableau élastique (pour faire plus classe, ne dites pas « élastique », dites « dynamique »), on écrira :

**Dim** MonTableau() **As** Integer

Ensuite, dès qu’on veut en fixer la taille, on écrit dans le code :

**Redim** MonTableau(11)

Si ce redimensionnement doit s’effectuer en conservant les valeurs précédemment entrées dans le tableau, on ajoutera le mot-clé **Preserve** :

**Redim Preserve** MonTableau(11)

Je déclare une constante, je suis la syntaxe suivante en remplaçant les nécessaires (Nom d’une constante et son type) :



**Exemple :**





1. **Je retiens l’essentiel.**

|  |
| --- |
| * Une variable est une entité dans laquelle on stocke les données nécessaires au déroulement du programme. * Le nom d’une variable est caractérisé comme suis :   + Les noms de variables n’ont pas de longueur maximale,   + Ils doivent commencer par un caractère,   + Ils ne doivent pas comporter d’espace,   + Ils ne sont pas sensibles à la casse (Toto et toto sont la même variable). * **Les types disponibles en VB sont:**   + **Type numérique**: 'Entier', 'réel' (**Integer**, Single) ;   + **Type alphanumérique**: 'Caractère' (Char) 'Chaîne de caractères'(String) ;   + **Booléen** : (Boolean soit 'Vrai' ou 'Faux') ;   + **Objet**. (Object) ;   + **Monétaire** (Décimal) ;   + **Date** (Date) ;   + **Matrice**, nombre imaginaire ;   + **Types complexes** (ou structurés) ;   + **Les Tableaux** (Array) ;   + **Les Collections**. * Pour déclarer une variable en VB on choisit d’abord l’emplacement de la déclaration soit globale ou locale en suite on suit la syntaxe qui suis :   **Dim** NomVariable **As** Type  La syntaxe de déclaration d’une constante est : |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
2. Définir une variable.
3. Donnez les éléments caractérisant une variable.
4. Donnez un exemple d’une variable.

**Rep1 :** une variable est une entité dans laquelle on stocke les données nécessaires au déroulement du programme.

**Rep1 :** Une variable est caractérisée par son nom et son type.

**Rep1 :** Un exemple d’une variable est : X2

* 1. **En utilisant mes connaissances**
     + 1. Donner la syntaxe de déclaration en VB.
       2. Que c’est qui caractérise le nom d’une variable ?

Rep1 la syntaxe de déclaration en VB est : **Dim** nom var **As** Type Var

Rep2. Le nom d’une variable est caractérisé comme suis :

* + Les noms de variables n’ont pas de longueur maximale,
  + Ils doivent commencer par un caractère,
  + Ils ne doivent pas comporter d’espace,
  + Ils ne sont pas sensibles à la casse (Toto et toto sont la même variable).
  1. **Pour aller plus loin**

Donnez une syntaxe pour déclarer un tableau « **bullitin** » qui recevra quinze noms des apprenants de votre classe.

Rep : **Dim** MonTableau(14) **As** string

[**https://fr.wikipedia.org/wiki/Visual\_Basic**](https://fr.wikipedia.org/wiki/Visual_Basic)**, lundi 15 octobre 2018**

[**https://www.scriptol.fr/programmation/basic.php**](https://www.scriptol.fr/programmation/basic.php) **, lundi 15 octobre 2018**

[**https://www.google.com/search?q=histoire+de+visual+basic&ie=utf-8&oe=utf-8**](https://www.google.com/search?q=histoire+de+visual+basic&ie=utf-8&oe=utf-8)

[**https://openclassrooms.com/fr/courses/993975-apprenez-a-programmer-en-vb-net/988743-historique-et-visual-basic-express-2010**](https://openclassrooms.com/fr/courses/993975-apprenez-a-programmer-en-vb-net/988743-historique-et-visual-basic-express-2010) **mardi 16 octobre 2018**

**Leçon n°6 : Distinction de types de modules en VB**.

1. **Je saurai:**

* Définir un module ;
* Différencier les types de modules utilisés en VB ;
* Déclarer un module en VB.

1. **J’observe et je réfléchi:**

|  |
| --- |
| **Image2.1** **Fenêtre montrant les différents éléments d’un projet « wimdowsApplication1 »**. |

**Activite1 : Observation**

J’observe les différents éléments d’un projet « wimdowsApplication1 ».

Je découvre qu’un contient plusieurs éléments dont les principaux sont : les forms, les modules, les classes.

**Activité 2 : Définir un module.**

Je défini un module comme une unité d'organisation contenant du code source donc son rôle est de regrouper du code utilisé dans plusieurs feuilles d’un projet.

**Activité 3 : Différencier les types de modules utilisés en VB ;**

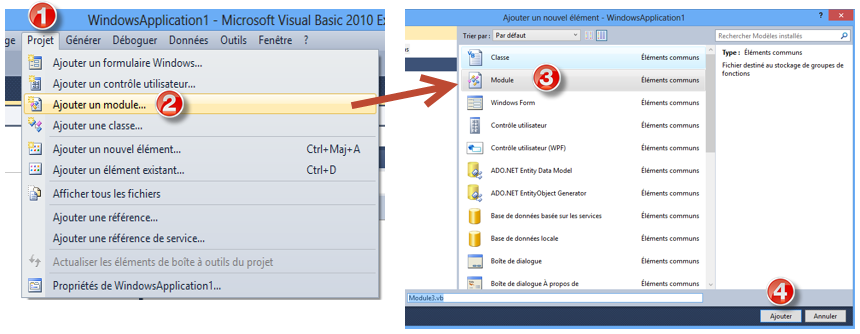
Je différencie les types de module:

* + - 1. **Le Module standard**

Le Module standard C'est un module qui **ne** contient **que** des déclarations, des procédures et des fonctions. Ce genre de module est celui qui se rapproche le plus de la notion de module réutilisable. Il contient du code qui n'est pas attaché spécifiquement à une application particulière. On se servira de tels modules lorsque l'on regroupera le code traitant un problème particulier, pour implanter un type abstrait de données.

Je donne une procédure pour créer un module standard sous MS VB 2010 Express

* Je clique sur le menu « **Projet »** (1) ;
* Je sélectionne «**ajouter un module** » (2);
* Je sélectionne « **Module** » (3)  et enfin je clique sur « **Ajouter** » (4). Figure ci-dessous.



* + - 1. **Le Module de feuille**

Ce type de module contient une description en codage interne à VB, d'une feuille des déclarations, des procédures et fonctions ainsi que des réactions de la feuille à certains événements programmés par le développeur. Un module de feuille sert à décrire une seule feuille, c'est le nom du module qui est le nom de la feuille.

Je donne une procédure pour créer un module de feuille sous MS VB 2010 Express

* Je clique sur le menu « **Projet »** (1) ;
* Je sélectionne «**ajouter un module** » (2);
* Je sélectionne « **WindowsForm** » (3)  et enfin je clique sur « **Ajouter** » (4).
  + - 1. **Le Module de classe**

C'est à ce niveau que VB nous permet de travailler avec les notions de base de la programmation orientée objet. Il est possible d'écrire du code de description de la classe représentée par le module de classe. Toutefois un tel module de classe ne peut pas contenir de contrôles (composants visuels), il est compilable séparément de l'application elle-même et peut être réutilisé. Un module de classe sert à décrire une seule classe, c'est le nom du module qui est le nom de la classe.

Je donne une procédure pour créer un module de classe sous MS VB 2010 Express

* Je clique sur le menu « **Projet »** (1) ;
* Je sélectionne «**ajouter un module** » (2);
* Je sélectionne « **Classe** » (3)  et enfin je clique sur « **Ajouter** » (4).
  + - 1. **Le Module Contrôle ActiveX**

Un contrôle ActiveX est un composant logiciel pouvant être intégré dans tout environnement de développement ou autre acceptant les contrôles ActiveX : Delphi et Visual Basic, ces navigateurs Internet tels que MS Internet Explorer et Netscape Navigator sont des exemples de tels environnements. Un contrôle ActiveX suit la spécification ActiveX dans le but de fournir aux programmeurs des composants logiciels réutilisables au sein d'applications et de services.   
La technologie ActiveX se différencie de la programmation orientée objets par son niveau d'intervention, c'est une technique de méta assemblage des objets :

* + La POO sert à **créer des objets** et en particulier à créer des composants ActiveX.
  + ActiveX est une norme technologique qui permet d'assembler des composants basés objets créés à l'aide d'outils de développement en particulier comme Visual Basic et Delphi, et beaucoup d'autres. Il s'agit d'une technologie ayant trait à l'**interopérabilité** **des objets**.

Je donne une procédure pour créer un module de classe sous MS VB 2010 Express

* Je clique sur le menu « **Projet »** (1) ;
* Je sélectionne «**ajouter un module** » (2);
* Je sélectionne « **Contrôle utilisateur** » (3)  et enfin je clique sur « **Ajouter** » (4).

NB : Tous les modules de VB comportent deux parties à savoir:

* une partie déclaration des **variables**, des **types** et **constantes**
* une partie déclaration et implémentation des procédures et des fonctions

1. **Je retiens l’essentiel.**

|  |
| --- |
| * Un module est une unité d'organisation contenant du code source donc son rôle est de regrouper du code utilisé dans plusieurs feuilles d’un projet. * **Le Module standard** C'est un module qui **ne** contient **que** des déclarations, des procédures et des fonctions. Ce genre de module est celui qui se rapproche le plus de la notion de module réutilisable * **Le module de feuille** contient une description en codage interne à VB, d'une feuille des déclarations, des procédures et fonctions ainsi que des réactions de la feuille à certains événements programmés par le développeur * **Le Module de classe**  permet de travailler avec les notions de base de la programmation orientée objet. Il est possible d'écrire du code de description de la classe représentée par le module de classe. * **Le Module** contrôle ActiveX est un composant logiciel pouvant être intégré dans tout environnement de développement ou autre acceptant les contrôles ActiveX : Delphi et Visual Basic. * Notez que tous les modules de VB comportent deux parties à savoir:   + une partie déclaration des **variables**, des **types** et **constantes ;**   + une partie déclaration et implémentation des procédures et des fonctions. |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
2. Définir un module.
3. Donnez sans expliquer les types des module utilise en VB.
   1. **En utilisant mes connaissances**
4. Donnez les étapes à suivre pour insérer un module en MS VB 2010 express.
5. Distinguez les parties d’un module en générale ?
6. **Pour aller plus loin**

Le langage VB dispose de 9 catégories d’objets ; Citez-les.

**Élément de la compétence : Ecrire un programme en Visual Basic.**

**Objectif pédagogique 1 : Utiliser les opérateurs.**

**Leçon 7 : Distinction des types des opérateurs.**

1. **Je saurai:**

* Définir un opérateur ;
* Différencier les types d’opérateurs utilisés en VB ;
* Utiliser les opérateurs en VB.

1. **J’observe et je réfléchi:**

|  |
| --- |
| **Image 7.1** **Fenêtre montrant les différents opérateurs** |

**Activite1 : Observation**

J’observe les différents opérateurs.

Je constate qu’en langage de programmation, on trouve plusieurs opérateurs de différentes fonctions.

**Activité 2 : Définir un opérateur en langage de programmation.**

Je défini un opérateur :

Les opérateurs sont des symboles qui permettent de manipuler des variables, c’est--dire effectuer des opérations, les évaluer.

**Activité 3 : Différencier les types des opéraeurs en VB ;**

* + - 1. **Opérateur d’affectation** « X est égal à …»

L’opération d’affectation se fait avec une signe d’égalité, voici la syntaxe :<Variable> = <expression>.

L’expression est évaluée puis affecter la valeur à une variable.

**Exemples : Dim X As integer**

**X=120**

* + - 1. **Opérateurs arithmétiques :** +, -, \*, /

L’addition, la soustraction, la multiplication et la division sont des opérateurs arithmétiques.

**Exemple :**

Dim A, B, C As Integer

B=10

C= 5

A=B-A «signifie que la valeur de A est 5»

A=B/5 « signifie que la valeur de A est 2 »

A=B+C « signifie que la valeur de A est 15 »

A= B\*C « signifie que la valeur de A est 50 »

L’opérateur modulo « % » permet d’obtenir le reste de la division entière. L’opérateur / retourne un quotient entier si les deux opérandes sont entiers.

**3. Opérateurs logiques**

&& : ET logique (and) **Exemple :** C && B =10 signifie **Faux** car C= 5

|| : OU logique (or) **Exemple :** C || B =10 signifie **Vrais** car B égale à 10

! : Négation logique (not) **Exemple :** C != B signifie **Vrais** parce que 5 est différent de 10

S’appliquent des expressions booléennes (0 si faux et valeur non nulle si vrai) ET retourne la valeur 1 si les deux opérandes sont non nuls, et 0 sinon. OU retourne la valeur 1 si au moins un des opérandes est non nul, et 0 sinon.

**4. Opérateurs de comparaison (Opérateurs relationnels)**

|  |  |
| --- | --- |
| < | opérande gauche inférieur à l'opérande droit  **EX :** C<B car 5<10 |
| <= | opérande gauche inférieur ou égal à l'opérande droit **EX :** C<=C car 5,00=5 |
| > | opérande gauche supérieur à l'opérande droit  **EX :** B>C car 10>5 |
| >= | opérande gauche supérieur ou égal à l'opérande droit  **EX :** B>=B car 10=10,00 |
| == | opérande gauche égal à l'opérande droit  **EX :** C==C car 5=5 |
| != | opérande gauche différent de l'opérande droit  **EX :** C !=B car 10≠5 |

Ces opérateurs retournent la valeur 0 si la comparaison est fausse et 1 sinon

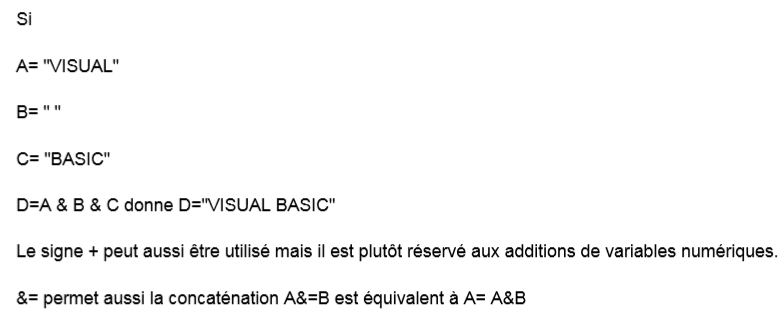
Exemple :

1. **Opérateurs d'assignation**

L’opérateur d'assignation permet de modifier les variables, les tableaux, les constantes et les objets de données.

|  |  |
| --- | --- |
| = | permet d'indiquer que l'opérande gauche est égal à la valeur de l'expression droite |
| += | Permet d'incrémenter l'opérande gauche de la valeur de l'expression droite  (x+=5) |
| -= | Permet de décrémenter l'opérande gauche de la valeur de l'expression droite (X-=5) |
| \*= | Permet de multiplier l'opérande gauche par la valeur de l'expression droite (x\*=5) |
| 0 /= | Permet de diviser l'opérande gauche par la valeur de l'expression droite (x/=5) |
| %= | Permet d'effectuer le modulo de l'opérande gauche par la valeur de l'expression droite  (x%2=1) |
| |= | Permet d'appliquer l'opérateur de bit OR à l'opérande de gauche avec la valeur de l'expression droite |
| &= | Permet d'appliquer l'opérateur de bit AND à l'opérande de gauche avec la valeur de l'expression droite |
| ^= | Permet d'appliquer l'opérateur de bit XOR à l'opérande de gauche avec la valeur de l'expression droite |
| <<= | Permet de décaler l'opérande gauche vers la gauche du nombre de bits spécifié par la valeur de l'expression droite |
| >>= | Permet de décaler l'opérande gauche vers la droite du nombre de bits spécifié par la valeur de l'expression droite |

Exemple de la concaténation



1. **Je retiens l’essentiel.**

|  |
| --- |
| * Les opérateurs sont des symboles qui permettent de manipuler des variables, c’est--dire effectuer des opérations, les évaluer. * Les différents types d’opérateurs utilisés en VB sont :  1. Opérateur d’affectation « =» 2. Opérateurs arithmétiques : « +, -, \*, / » 3. Opérateurs logiques : « &&,||, ! » 4. Opérateurs de comparaison : « <,>,<=,>=, !=, <> » 5. Opérateurs d'assignation : « +=, -=, =/,\*=,%=,!= ,&=, ^=, <<=,>>= » |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
2. Définir un opérateur.
3. Donnez sans expliquer les opérateurs logiques utilisés en VB.

**Rep1 :** Les opérateurs sont des symboles qui permettent de manipuler des variables, c’est--dire effectuer des opérations, les évaluer.

**Rep2 :** les opérateurs logiques utilisés en VB sont :

* && : ET logique (and) ;
* || : OU logique (or) ;
* ! : Négation logique (not).
  1. En utilisant mes connaissances

Soit l’instruction : **Dim X, Y, Z, T As integer. Si X=Y=5, Z=3 et T=1.**

Donnez, Sous MS VB 2010 Express les valeurs des expressions suivantes.

* 1. H=X\*Y+T
  2. F= X/Z
  3. J=F-H
  4. K=J^T

Rep :

* 1. H=X\*Y+T=5\*5+1=26
  2. F= X/Z=5/3=1.666
  3. J=F-H=1666-26=-24.334
  4. K=J^T=-24.334^1=-24.334
  5. Pour **aller plus loin**

Sous MS VB 2010 Express, Ecrire un programme qui demande un apprenant d’entrer son année de naissance est qui affiche son âge.

**Rep :**

[**https://rmdiscala.developpez.com/cours/LesChapitres.html/Cours5/Chap5.9.htm#Mstandard**](https://rmdiscala.developpez.com/cours/LesChapitres.html/Cours5/Chap5.9.htm#Mstandard)

[**http://www.info-3000.com/vbvba/modulesdeclasse.php**](http://www.info-3000.com/vbvba/modulesdeclasse.php)

**Leçon 8 : Utilisation des variables, des opérations et constantes.**

1. **Je saurai:**

* Déclarer les variables et les constantes;
* Utiliser les variables, les opérations et constantes.

1. **J’observe et je réfléchi:**

|  |
| --- |
| Image 9.1 les syntaxes de déclaration de différentes variables, types et niveaux |

**Activite1 : Observation**

Je constate qu’en langage de programmation Visual Basic, on trouve plusieurs méthodes de déclaration suivant l’utilisation de la variable.

Le mot clé **const** permet de déclarer une variable de type constante.

**Activité 2 : Déclarer une variable et une constante;**

Pour déclarer en VB, on utilise les syntaxes suivantes :

Dim A,B, C As Integer ' déclaration des variables A,B,C de type entier.

* Pour déclarer une variable nous utilisons le mot clé **Dim** suivi d’un nom de variable puis le mot clé **As** et enfin le type de données de cette variable.

Public Const c As Double = 3.14 ' déclaration d’une constante pi

Const k As AccessibleEvents = 5 'déclaration d'une variable des événements nommé k de la valeur est 5

**Je donne quelques consignes à respecter pour bien utiliser les variables et les constantes en programmation VB**

* une variable sert à stocker un nombre, du texte (chaîne de caractères), une date, un objet.
* On peut utiliser dans les noms de variable des majuscules ou des minuscules mais en VB il n'y a pas de différence.
* On ne peut pas déclarer une variable B et une variable b cause l’ambiguïté.
* Si on déclare une variable nommée 'B' et si ultérieurement on tape 'b', le logiciel le transforme automatiquement en 'B'.
* On peut mettre des chiffres et des lettres dans les noms de variable mais pas de chiffres en premier caractère. 2A n'est pas un nom valide Nom2 l'est.
* Certains caractères de ponctuation ('..) ne peuvent pas être utilisés, d'autres "\_" et "||" sont permis comme exemple **«nom\_Utilisateur»** est valide. L'espace n'est pas permis.
* Les mots clé de VB ne peuvent pas être utilisés comme les noms des variables : On ne peut pas nommer une variable **Form, private, sub, class** ou **BackColor**
* On ne peut pas affecter une variable d'un type à une variable d'un autre type:
* si A est numérique et B est alphanumérique (chaîne de caractères) A=B n'est pas accepté.
* L'affection marche pour les objets, leurs propriétés.
* Bouton1.BackColor= Bouton2.BackColor
* Signifie que l'on donne au Bouton1 la même couleur de fond que celle du bouton2: on affecte la valeur BackColor du Bouton2 au Bouton1.
* On peut forcer le type d'un littéral en ajoutant une lettre:
  + i= 42L 'le 'L' indique que 42 est un Long
  + i= 42I 'le 'I' indique que 42 est un Integer
  + i= 42D 'le 'D' indique que 42 est un Decimal
  + i= 42S 'le 'S' indique que 42 est un Single
  + C = "A"c 'le 'c' force "A" à être une Char et non une String

**Je donne quelques consignes à respecter pour bien utiliser les opérateurs en programmation VB.**

* En VB comme en mathématique les opérateurs permettent delier deux éléments (variables ou objets) et de fournir un résultat
* **Pour travailler sur les variables on utilise des opérateurs (addition « + », soustraction «- », la multiplication « \* », la division « / » et Modulo).**

Exemple dans le cas de variables numériques.

Dim A,B, C As Integer

B=2

C=3

A=B+C ‘ signifie que A est égale à 5

A=B^C ‘ signifie que A est égale à 8

A=C\*B ‘ signifie que A est égale à 6

A=C/B ‘ signifie que A est égale à 1,5

B Mod 3 donne 1‘ signifie que B n’est pas divisible de 3

A Mod 3 = 0, signifie que A est un multiple de 3

**TextBox3.**Text = Val(textbox1.Text) \* (texbox2.Text) ‘signifie que TextBox3.Text est égale au produit de la valeur de textbox1 et texbox2.

**Activité 3 : Exemple compilé en MS VB 2010 Exprès.**

Sous MS VB 2010 Exprès, Réaliser une calculatrice à quatre opérations opérées sur deux nombres entrés au clavier.

Je lance MS VB 2010 Exprès, je crée un projet dans lequel j’entre les codes suivants :

|  |  |
| --- | --- |
|  |  |
| Opération de la divion | Opération de la multiplication |

Public Class Form1

Private Sub TextBox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles textbox1.TextChanged

textbox1.Text = textbox1.Text

If Not IsNumeric(textbox1.Text) Then

MsgBox("entrer le nombre seulemnt!")

End If

End Sub

Private Sub TextBox2\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles texbox2.TextChanged

texbox2.Text = texbox2.Text

If Not IsNumeric(texbox2.Text) Then

MsgBox("entrer le nombre seulemnt!")

End If

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Button2.Text = "+"

TextBox3.Text = Val(textbox1.Text) + (texbox2.Text)

texbox2.Text = texbox2.Text

End Sub

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

Button1.Text = "\*"

TextBox3.Text = Val(textbox1.Text) \* (texbox2.Text)

End Sub

Private Sub Button3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click

Button3.Text = "-"

TextBox3.Text = Val(textbox1.Text) - (texbox2.Text)

End Sub

Private Sub Button4\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click

Button1.Text = "\*"

TextBox3.Text = Val(textbox1.Text) / (texbox2.Text)

End Sub

Private Sub Button5\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click

Button5.Text = "Annuler"

textbox1.Text = ""

texbox2.Text = ""

TextBox3.Text = ""

End Sub

Private Sub TextBox1\_TextChanged\_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles textbox1.TextChanged

textbox1.Text = textbox1.Text

If Not IsNumeric(textbox1.Text) Then

MsgBox("Enttrer nombre seulement")

End If

End Sub

Private Sub TextBox3\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox3.TextChanged

End Sub

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

End Sub

End Class

1. **Je retiens l’essentiel.**

|  |
| --- |
| * une variable sert à stocker un nombre, du texte (chaîne de caractères), une date, un objet. * Certains caractères de ponctuation ('..) ne peuvent pas être utilisés, d'autres "\_" et "||" sont permis comme exemple **«nom\_Utilisateur»** est valide. L'espace n'est pas permis. * Les mots clé de VB ne peuvent pas être utilisés comme les noms des variables : On ne peut pas nommer une variable **Form, private, sub, class** ou **BackColor** * On ne peut pas affecter une variable d'un type à une variable d'un autre type: * si A est numérique et B est alphanumérique (chaîne de caractères) A=B n'est pas accepté. * L'affection marche pour les objets, leurs propriétés.   Ex : Bouton1.BackColor= Bouton2.BackColor   * On peut forcer le type d'un littéral en ajoutant une lettre:   + i= 42L 'le 'L' indique que 42 est un Long   + i= 42I 'le 'I' indique que 42 est un Integer   + i= 42D 'le 'D' indique que 42 est un Decimal   + i= 42S 'le 'S' indique que 42 est un Single   + C = "A"c 'le 'c' force "A" à être une Char et non une String |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
2. Écrire le code créant une variable nommée **'myNumber'** devant contenir un entier, donner lui la v aleur 12. Nommer en terme Visual Basic les différents éléments et étapes.
3. Voici des déclarations de variables, indiquer les noms de variables corrects et ceux qui ne seront pas acceptés.

Dim 2a As Integer

Dim maPremiereVariableDeBoucleInterne As Integer

Dim nom Utilisateur As String

Dim MonNom As String

**Rep1 :** Dim myNumber As Integer =12

*On aurait pu aussi écrire:*

Dim myNumber As Integer

myNumber = 12

**Rep2 :**

* Dim 2a As Integer ‘Erreur: un nom de variable ne doit pas commencer par un chiffre.
* Dim maPremiereVariableDeBoucleInterne As Integer ‘ Correcte: le nom d'une variable peut être très long.
* Dim nom Utilisateur As **Integer** ‘Erreur: un nom de variable ne doit pas contenir d'espace.
* Dim MonNom As String ‘Correcte et accepté mais on aurait pu écrire 'monNom' pour suivre les règles de bonnes écritures qui consistent à mettre en majuscules la première lettre de chaque mot sauf pour le premier mot.
  1. **En utilisant mes connaissances**

1. Quel type de variable utilisés pour :

- Faire des calculs financiers?

- Mettre un texte de 500 caractères?

- Une variable de boucle allant de 0 à 100 et qui soit la plus rapide possible?

2. Déclarer une constante qui se nomme myName et lui donner la valeur "lulu".

Rep2.

* + 1. Les types de variable utilisés pour :
* Faire des calculs financiers? Les Decimal.
* Mettre un texte de 500 caractères? une String.
* Un variable de boucle allant de 0 à 100 et qui soit le plus rapide possible? un Integer.

1. Déclarer une constante qui se nomme myName et lui donner la valeur "lulu"
   1. **Pour aller plus loin**

Créer une variable s de type String, l'initialiser avec "Philippe", afficher dans une MessageBox la longueur de la chaîne

Dim s As String ="Philippe"

MsgBox(s.Length.ToString) ‘ s.length retournant un entier, il faut le transformer en chaîne de caractères (grâce à .ToString) pour l'afficher

**Objectif pédagogique 2 : connaitre les instructions VB**

**Leçon 9 : Distinction des structures des contrôles.**

1. **Je saurai:**

* Définir chaque structure de contrôle ;
* Distinguer les différentes structures de contrôle ;
* Expliquer le mode de fonctionnement de chacun structure.

1. **J’observe et je réfléchi:**

|  |
| --- |
| Image2.1 Fenêtre montrant les différentes structures de contrôle |

**Activite1 : Observation**

Je constate qu’en langage de programmation, on trouve les différentes structures de contrôle.

**Activité 2 : Définir une structure de contrôle.**

En programmation informatique, une structure de contrôle est une instruction particulière d'un langage de programmation impératif pouvant dévier le flot de contrôle du programme la contenant lorsqu'elle est exécutée.

Si, au plus bas niveau, l'éventail se limite généralement aux branchements et aux appels de sous-programme, les langages structurés offrent des constructions plus élaborées comme :

* **les conditionnelles**  (if, if…else, switch),
* **les itératives** (while, do…while, for), ou encore les appels de fonction.

Outre ces structures usuelles, la large palette des structures de contrôle s'étend des constructions de gestion d'exceptions (try–catch, fréquemment trouvés dans les langages de haut niveau aux particularismes de certains langages comme les instructions différées (defer) de Go.

**Activité 3. Distinguer les différentes structures de contrôle**

On distingue les structures de contrôle **conditionnelles** et **itératives**.

**Activité 4. Expliquer le mode de fonctionnement de chacun structure.**

* + 1. **Les structures de contrôle conditionnelles :**

**J’explique le fonctionnement de la structure conditionnelle (Si ... Alors ...)**

La conditionnelle permet d’exécuter une séquence d’instruction, seulement si une condition est vraie.

if condition:

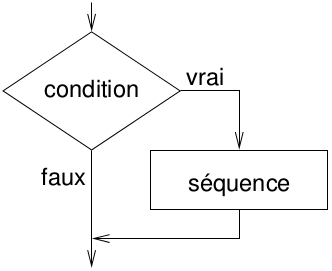
sequence

suite

La condition si est nécessairement une expression booléenne.

Évaluation :

* la condition est évaluée;
* si la condition est vraie, la séquence est exécutée puis le contrôle passe à la suite;
* si la condition est fausse, le contrôle passe à la suite, sans exécuter la séquence.

[](http://python-liesse.enseeiht.fr/cours/_images/Organigramme-SiAlors.png)

**J’explique le fonctionnement de la structure conditionnelle (Si...Alors...Sinon...)**

Cette structure permet de poser une alternative.

if condition:

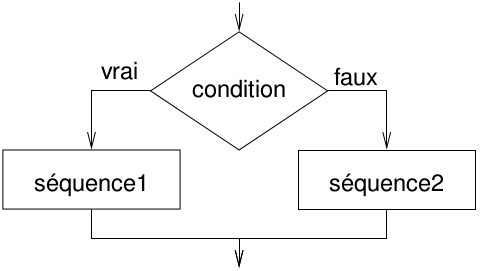
sequence1

else:

sequence2

suite

Si la condition est vraie, c’est la séquence1 qui sera exécutée, sinon c’est la séquence2 qui sera exécutée. Dans tous les cas, l’exécution continuera à suite.

[](http://python-liesse.enseeiht.fr/cours/_images/Organigramme-SiSinon.png)

**J’explique le fonctionnement de la structure conditionnelle Si imbriquée.**

Naturellement, il est possible d’imbriquer les structures de contrôle conditionnelles les unes à l’intérieur des autres, comme ceci :

if condition1:

if condition2:

sequence

suite

Toutefois, on préfèrera l’utilisation d’une conjonction dans la condition (car les opérateurs booléens s’exécutent en court-circuit) :

if condition1 and condition2:

sequence

suite

**J’explique le fonctionnement de la clause SinonSi**

On peut ajouter autant de clause SinonSi (elseif) que de conditions à tester si les précédentes ont renvoyé faux.

if condition1:

sequence1

elif condition2:

sequence2

elif condition3:

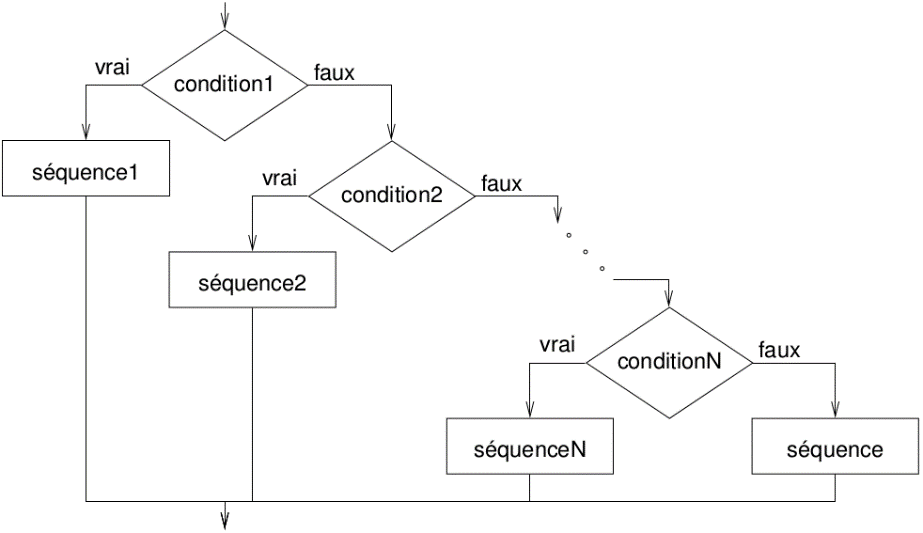
sequence3

else:

sequence4

suite

Évaluation : Les conditions sont évaluées dans l’ordre. Si une condition s’évalue à vraie, la séquence associée est exécutée puis l’exécution continue à suite, sinon la condition suivante est envisagée. La partie else n’est donc exécutée que si aucune des conditions ne s’est évaluée à vrai.

[](http://python-liesse.enseeiht.fr/cours/_images/Organigramme-SiSinonSi.png)

* + 1. **Les structures de contrôle itératives (Répétitives)**

Les répétitions permettent d’exécuter plusieurs fois une même séquence d’instructions. Il est important de vérifier qu’une répétition se termine effectivement (notion de variant).

**J’explique le fonctionnement de la répétition (boucle) Tant Que**

La répétition TantQue permet d’exécuter une séquence d’instructions tant qu’une condition est vraie.

Évaluation :

1. La condition est évaluée.
2. Si la condition est vraie, on exécute la séquence et on recommence en 1.
3. Si la condtion est fausse, on exécute la suite (on sort donc de la boucle).

|  |  |
| --- | --- |
| while condition:  sequence  suite | [Organigramme TantQue](http://python-liesse.enseeiht.fr/cours/_images/Organigramme-TantQue.png) |

**Attention** : afin de garantir la terminaison du programme, il faut s’assurer que la condition va finir par être fausse, et ainsi que le programme ne restera pas infiniment dans la boucle. C’est pour cela que la séquence d’instructions doit modifier la condition.

**J’explique le fonctionnement de la boucle « jusqu’à ce que » à postcondition**

|  |  |
| --- | --- |
| REPETE  Instruction 1  JUSQUACEQUE Condition 1  Instruction 2 | [Cf-until-fr.svg](https://commons.wikimedia.org/wiki/File:Cf-until-fr.svg?uselang=fr) |

Cette boucle permet de réitérer une instruction ou un ensemble d'instructions jusqu'à ce qu'une condition de sortie soit vérifiée. La série d'instructions est exécutée au moins une fois, quelle que soit la condition.

**J’explique le fonctionnement de la boucle *for***

Ce type de boucle, que l'on retrouve fréquemment, permet de condenser :

* une initialisation ;
* une condition ;
* une incrémentation.

On retrouve sur la ligne du for les 3 instructions Ex : For (i=0 ; i<10 ;i++)

* Une initialisation (compteur = 0) : la variable compteur est mise à 0 au tout début de la boucle.
* Une condition (compteur < 10) : on vérifie que la variable compteur est inférieur à 10 à chaque nouveau tour de boucle.
* Une incrémentation (compteur ++) : à chaque tour de boucle, on ajoute 1 à la variable compteur! donc on voit s'afficher à l'écran dix fois de numéro 0 à 9.

On peut faire autre une incrémentation si on le désire. La dernière section du for est réservée à la modification de la variable et on peut donc y faire une décrémentation (compteur--) ou avancer de n en n (compteur += n). Ex : (for x=1 ; x<15 ; x+n)

Note qu'il est courant d'initialiser la variable directement à l'intérieur du for, comme

La variable n'existe alors que pendant la durée de la boucle for. C'est la forme la plus courante de cette boucle. On ne déclare la variable avant le for que si on en a besoin plus tard, ce qui est un cas assez rare.

**Je compare les cas d’utilisation des structures de contrôle.**

* Les conditions permettent de tester la valeur des variables et de modifier le comportement du programme en conséquence.
* La condition de type if (si) ...else if (sinon si) ...else (sinon) est la plus courante.
* La condition switch, plus spécifique, permet de tester les différentes valeurs possibles d'une seule variable.
* Les boucles permettent de répéter les mêmes instructions plusieurs fois.
* On distingue trois types de boucles : while,do... while et for.
* La boucle for est généralement utilisée lorsqu'on sait combien de fois on souhaite répéter les instructions, tandis que while et do... while sont plutôt utilisées lorsqu'on souhaite répéter des instructions jusqu'à ce qu'une condition spécifique soit vérifiée.

1. **Je retiens l’essentiel.**

|  |
| --- |
| En programmation informatique, une structure de contrôle est une instruction particulière d'un langage de programmation impératif pouvant dévier le flot de contrôle du programme la contenant lorsqu'elle est exécutée.  On distingue les structures de contrôle conditionnelles et itératives (répétitives).  Les structures de contrôle conditionnelles permettent d’exécuter une séquence d’instruction, seulement si une condition est vraie.  Les structures de contrôle répétitives permettent d’exécuter plusieurs fois une même séquence d’instructions. Il est important de vérifier qu’une répétition se termine effectivement (notion de variant).  ***For*** *est une boucle qu’on retrouve fréquemment et qui permet de condenser 3 instructions à une seule ligne à savoir:*  une initialisation ;  une condition ;  une incrémentation. |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
2. Définir une boucle.
3. Donnez sans expliquer les différentes structures de contrôle conditionnelles
4. Quelle est la structure de contrôle à choisir pour entre un nombre positif non null ?

**Rep1 :** Les opérateurs sont des symboles qui permettent de manipuler des variables, c’est--dire effectuer des opérations, les évaluer.

**Rep2 :** les différentes structures de contrôle conditionnelles sont :

* La condition **Si** d’une expression booléenne (Si …..Alors …..)
* La condition **Si** qui permet de poser une alternative (Si...Alors...Sinon...)
* La condition **Si** imbriqué (Si cond1 et Si cond2 Alors …..)
* La condition **Si** de la clause (Si...Alors...Sinon si... Sinon si...Sinon si...Sinon…..)

**Rep3 :** pour entre un nombre positif non null on doit utiliser un boucle répétitif tel que: tantque... Faire… ou répéter… jusquà…

* 1. **En utilisant mes connaissances**

1. A l’aide d’un schéma expliquer le fonctionnement d’une boucle tant que …faire….
2. L’enseignant demande aux apprenants de spécifier une structure de contrôle à utiliser pour créer un programme permettant de réaliser l’attribution de diplôme comme suis :

* Au moins 85% = « Plus Grande Distinction » ;
* Entre 85% et 70%=  « Grande Distinction » ;
* Entre 70% et 50%=  « Distinction » ;
* Entre 60% et 50%=  « Satisfaction » ;
* Au plus 49% =« Exclus d’office ».

Rep1 :

|  |  |
| --- | --- |
| [Organigramme TantQue](http://python-liesse.enseeiht.fr/cours/_images/Organigramme-TantQue.png) | La condition est évaluée comme suis :  Si la condition est vraie, on exécute la séquence et on recommence en 1.  Si la condition est fausse, on exécute la suite (on sort donc de la boucle). |

Rep2.

Pour telle programme, on préfère une structure de contrôle conditionnel de clause car c’est elle qui exécute plusieurs conditions de suite hiérarchique.

if condition1:

sequence1

elif condition2:

sequence2

elif condition3:

sequence3

else:

sequence4

suite

* 1. **Pour aller plus loin**

Quel est le rôle de la commande [**MsgBox ("……..")**] en VB

Rep : La commande [MsgBox (" ……")] en VB permet d’afficher un message qui est entre guillemet dans une boite de dialogue.

**Leçon 10 : Utilisation du structure de contrôle conditionnelle Si (If).**

1. **Je saurai:**

* Déclarer les variables, les opérations et constantes;
* Utiliser les différentes structures de contrôle de la condition IF.

1. **J’observe et je réfléchi:**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Image 11.1 Fenêtre montrant les différentes structures de la condition IF** | | |

**Activité 1 : Observation**

Je constate qu’une structure de contrôle conditionnelle peut avoir les différentes formes.

**Les structures de contrôle à choix** permettent de gérer le déroulement du code. If … Then et de créer une structure décisionnelle c’est à dire qu’une instruction (ou un bloc d'instructions) peut être exécutée si une condition est vraie.

**Activité 2 : Utilisation de la condition IF … Then**

|  |  |
| --- | --- |
| If Condition Then  End if  Si la Condition est vraie alors... | **Exemple**:  If( A=B) then  MsgBox("A est égale à B")  End If  Si A = B alors on exécute le bloc de code qui se situe entre **Then** et **End** **If**, il affiche dans une fenêtre MessageBox «A est égale à B» |

**Noter que, on peut écrire sur la même ligne après Then dans ce cas (Pas besoin de End If).**

If (A=B) Then MsgBox("A est égale à B")

On peut meme tester une condition fausse et dans ce cas on utilise le mot clé **Not**.

If Not (A=B) Then MsgBox("A est différent de B")

Si A et B sont différent (Not A=B signifie que A et B ne sont pas égaux) dans ce cas on affiche "A est différent de B".

On aurait pu écrire If (A<>B) Then MsgBox ("A est différent de B")

If Not IsNumeric(n) then MsgBox ("n n'est pas un nombre")

Exit Sub End if

Si n n'est pas numérique alors afficher dans une boite de dialogue: « n n'est pas un nombre » puis quitter la procédure (Exit Sub)

**Noter** bien que comme il y a plusieurs instructions après Then on crée un bloc d'instruction de plusieurs lignes entre **Then** et End **If**.

**Simplification d'écriture:**

|  |  |
| --- | --- |
| Au lieu de  If Condition = True Then … End if | On peut écrire:  If Condition Then … End if |

Condition étant de toute manière évaluée pour voir si elle est égale à True.

**Activité 3 : Utiliser la structure Si...Alors...Sinon…**

If condition then … effectué si condition vraie … Else… effectué si condition fausse... End if

**Exemple**:

If (A=B) then MsgBox("A est égale à B")

Else

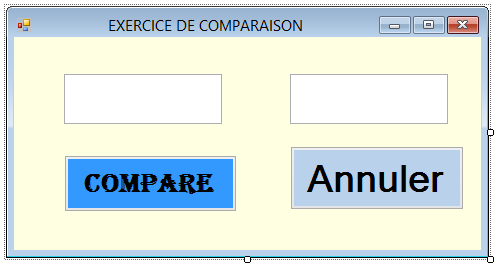
MsgBox ("A différent de B") End If

**Activité 4 : Exemple pratique :**

Ecrire un programme en **MS VB 2010 Express** permettant d’entrer deux entiers, de les comparer et d’afficher : « A est égale à B » s’ils sont égaux ou « A est différents de B » dans le cas contraire.

**Je donne les étapes de résolution:**

* + - 1. Sous MS VB 2010 Express, Je crée l’interface suivante :



* + - 1. J’entre dans la fenêtre des codes et je tape les codes suivants:

Public Class Form1

Public A, B As Integer

Private Sub TextBox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles textbox1.TextChanged

textbox1.Text = textbox1.Text

If Not IsNumeric(textbox1.Text) Then

MsgBox("Entrer un entier!")

End If

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

B = texbox2.Text

A = textbox1.Text

Button2.Text = "COMPARE"

If (A = B) Then

MsgBox("A est égal à B")

Else

MsgBox("A est différent de B")

End If

End Sub

Private Sub texbox2\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles texbox2.TextChanged

texbox2.Text = texbox2.Text

If Not IsNumeric(texbox2.Text) Then

MsgBox("Entrer un entier!")

End If

End Sub

End Class

* + - 1. Après avoir écrit les codes, je compile mon programme pour voir le résultat en cliquant sur « démarrer le débogage ou en appuyant sur **F5**»
      2. Je tape les nombres à comparer et je clique sur le bouton «comparer», on aura les interfaces présentées de façon suivante :

|  |  |
| --- | --- |
|  |  |

**Activité 5 : Utilisation des structures If Then peuvent être imbriquées :**

En VB, Il peut y avoir des opérateurs logiques dans la condition IF… Then…:

**Ex**: If (A=B) And (C=D) then… Cela signifie Si A égal B et si C égal D alors …

**Ex** : If (A=B) Ou (C=D) then… Cela signifie Si A égal B et si C égal D alors …

Pour tester si la valeur est numérique, on écrit:

If … If… Else ... If… End if … End if … End If

Pour bien repérer les différents niveaux, utiliser les tabulations et décaler le 'If … then' et son code au même niveau.

Pour vérifier s'il n'y a pas d'erreur, compter les 'If' il doit y en avoir autant que des 'End If'. VB souligne le 'If' si il n'y a pas de 'End if'.

If Condition1 Then... Else If *condition2* Then... ElseIf **condition3** Then ... end if

Si *condition1*...

Sinon si *condition2*

Sinon si *condition3*

Fin Si

**Je retiens l’essentiel.**

|  |
| --- |
| **Les structures de contrôle à choix** permettent de gérer le déroulement du code. If … Then et de créer une structure décisionnelle c’est à dire qu’une instruction (ou un bloc d'instructions) peut être exécutée si une condition est vraie.  Les différentes formes de la condition IF  1ère forme  If( condition) then  Instruction  End If  2ième forme  If Condition = True Then Instruction  End if  3ième forme  If Condition Then Instruction1  Else  Instruction2  End If  4ième forme  If **Condition1** Then **instruction1**  Else If ***condition2*** Then **instruction2**  ElseIf **condition3** Then **instruction3**  end if  5ième forme  If condition1 then instruction1  If condition2 then instruction2  End If  End If  6ième forme  If ((condition1) and (Condition2)) then instruction1  Else instruction2   * End If |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
2. Ecrire une syntaxe de la condition If qui admet deux réponses alternatives.
3. Ecrire une syntaxe de la condition If qui admet deux réponses alternatives de deux conditions combines.

**Rep1 :** If Condition Then Instruction1

Else

Instruction2

End If

**Rep1 :**

If ((Condition1) & (Condition2)) Then Instruction1

Else

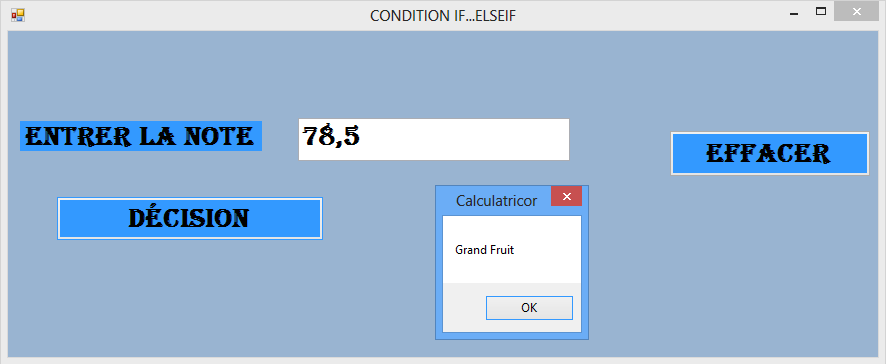
Instruction2

End If

* 1. **En utilisant mes connaissances**

1. Ecrire un programme permettant d’entrer le pourcentage de la note d’un apprenant et afficher la décision comme suit :

* Plus Grand Fruit si le pourcentage de la note dépasse 80 ;
* Grand Fruit si le pourcentage de la note est entre 80 et 70 ;
* Fruit si le pourcentage de la note est entre 70 et 60 ;
* Satisfaction si le pourcentage de la note est entre 60 et 50 ;
* Exclut d'office si le pourcentage de la note est inférieur à 50.



Public Class Form4

Dim N As Integer

Private Sub texbox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged

TextBox1.Text = TextBox1.Text

If Not IsNumeric(TextBox1.Text) Then

MsgBox("Entrer un entier!")

End If

End Sub

Private Sub Label1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label1.Click

Label1.Text = Label1.Text

End Sub

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

N = TextBox1.Text

Button1.Text = "Décision"

If (N > 80) Then

MsgBox("Plus Grand Fruit")

ElseIf (N > 70) Then

MsgBox("Grand Fruit")

ElseIf (N > 60) Then

MsgBox("Fruit")

ElseIf (N >= 50) Then

MsgBox("Satisfaction")

Else

MsgBox("Exclut d'office")

End If

End Sub

Private Sub Button2\_Click\_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Button2.Text = Button2.Text

TextBox1.Text = ""

End Sub

End Class

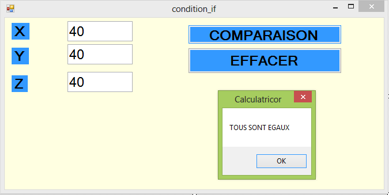
1. Ecrire un programme permettant d’entrer trois entiers et d’afficher comme suit :

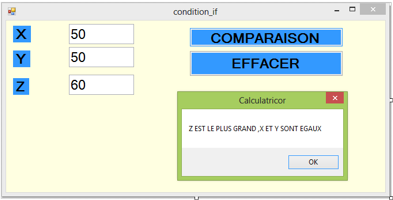
…. est le plus grand,

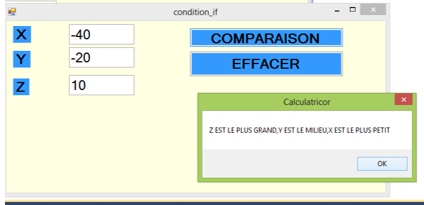
…. est le plus petit,

…..est l’intermédiaire de la liste. (Avec X, Y, Z les nombres entiers entrés au clavier).

**Figure ci-après :**







Rep

Public Class condition\_if

Dim X, Y, Z As Integer

Private Sub TextBox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged

TextBox1.Text = TextBox1.Text

End Sub

Private Sub TextBox2\_TextChanged\_1(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox2.TextChanged

TextBox2.Text = TextBox2.Text

End Sub

Private Sub TextBox3\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox3.TextChanged

TextBox3.Text = TextBox3.Text

End Sub

Private Sub Label1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label1.Click

Label1.Text = Label1.Text

End Sub

Private Sub Label2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label2.Click

Label2.Text = Label2.Text

End Sub

Private Sub Label3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label3.Click

Label3.Text = Label3.Text

End Sub

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

X = TextBox1.Text

Y = TextBox2.Text

Z = TextBox3.Text

Button1.Text = "COMPARAISON"

If ((X = Y) And (Z > X)) Then

MsgBox("Z EST LE PLUS GRAND ,X ET Y SONT EGAUX")

End If

If ((X > Y) And (Y = Z)) Then

MsgBox("X EST LE PLUS GRAND,Y ET Z SONT EGAUX")

End If

If ((X = Z) And (Y > Z)) Then

MsgBox("Y EST LE PLUS GRAND,X ET Z SONT EGAUX")

End If

If ((X = Z) And (X > Y)) Then

MsgBox("Y EST LE PLUS PETIT ,X ET Z SONT EGAUX")

End If

If ((Y = Z) And (Z > X)) Then

MsgBox("X EST LE PLUS PETIT ,Y ET Z SONT EGAUX")

End If

If ((X = Y) And (X > Z)) Then

MsgBox("Z EST LE PLUS PETIT ,X ET Y SONT EGAUX")

End If

If ((X > Y) And (X > Z) And (Y > Z)) Then

MsgBox("X EST LE PLUS GRAND,Y EST LE MILIEU,Z EST LE PLUS PETIT")

ElseIf ((X > Y) And (X > Z) And (Y < Z)) Then

MsgBox("X EST LE PLUS GRAND,Z EST LE MILIEU,Y EST LE PLUS PETIT")

End If

If ((Y > Z) And (Y > X) And (Z > X)) Then

MsgBox("Y EST LE PUS GRAND,Z EST LE MILIEU,X EST LE PLUS PETIT")

ElseIf ((Y > Z) And (Y > X) And (Z < X)) Then

MsgBox("Y EST LE PLUS GRAND,X EST LE MILIEU,Z EST LE PLUS PETIT")

End If

If ((Z > X) And (Z > Y) And (Y > X)) Then

MsgBox("Z EST LE PLUS GRAND,Y EST LE MILIEU,X EST LE PLUS PETIT")

ElseIf ((Z > X) And (Z > Y) And (Y < X)) Then

MsgBox("Z EST LE PLUS GRAND,X EST LE MILIEU,Y EST LE PLUS PETIT")

End If

If ((X = Y) And (Y = Z)) Then

MsgBox("TOUS SONT EGAUX")

End If

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Button2.Text = "EFFACER"

TextBox1.Text = ""

TextBox2.Text = ""

TextBox3.Text = ""

End Sub

End Class

* 1. **Pour aller plus loin**

Combien d’instruction au minimum qu’une structure de contrôle conditionnelle peut supporter ?

**Rep :** une structure de contrôle conditionnelle peut supporter  une seul instruction au minimum.

**Leçon 11 : Utilisation de la structure de contrôle Select Case**

**Je saurai:**

* Reconnaitre la syntaxe de la structure de contrôle select case ;
* Utiliser la structure de contrôle select case ;
* Distinguer les mots clés d’une structure de contrôle select case.

1. **J’observe et je réfléchi:**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Image 12.1 Fenêtre montrant le fonctionnement de la structure de contrôle Select Case** | | |

**Activité 1 : Observation**

Je constate qu’une structure de contrôle select case peut avoir plusieurs alternative une fois la condition est réalisée.

**Activité 2 : Reconnaitre la syntaxe de la structure de contrôle select case**

**Select Case**

La structure de contrôle **Select Case** permet de créer une structure décisionnelle permettant d'exécuter un grand nombre de blocs de code différents en fonction de la valeur d'une expression.

**Select Case** expression

**Case** valeur1 code effectué si expression=valeur1

**Case** valeur2 code effectué si expression=valeur2

**Case** valeur3 code effectué si expression=valeur3

..

**Case** Else code effectué dans tous les autres cas

End Select

**Activité 3 : Utiliser la structure de contrôle select case**

**Exemple :** Ecrire un programme permettant d’afficher le jour de la semaine avec n, n étant un entier contenant le numéro d'ordre du jour.

|  |  |
| --- | --- |
|  |  |

Public Class Form1

Public N As Integer

Private Sub Label1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label1.Click

End Sub

Private Sub TextBox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged

TextBox1.Text = TextBox1.Text

If Not IsNumeric(TextBox1.Text) Then

MsgBox("entrer un entier SVP")

End If

End Sub

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

Button1.Text = "Nom du jour"

N = TextBox1.Text

Select Case N

Case 1

MsgBox("Lundi")

Case 2

MsgBox("Mardi")

Case 3

MsgBox("Mercredi")

Case 4

MsgBox("Jeudi")

Case 5

MsgBox("Vendredi")

Case 6

MsgBox("Samedi")

Case 7

MsgBox("Dimanche")

Case Else

MsgBox(" ce n'est Pas le jour de la semaine")

End Select

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Button2.Text = "Effacer"

TextBox1.Text = ""

End Sub

End Class

Avec la structure de contrôle **Select Case**, plusieurs clauses d'expression peuvent être séparées par des virgules.

Select Case N

Case 8, 9, 10 'Effectuer le code si N=8 ou N=9 ou N=10

**Activité 4 : Distinguer les mots clés d’une structure de contrôle select case**

* + 1. Le mot clé **To** permet de définir les limites d'une plage de valeurs correspondantes pour N.

Select Case N

Case 8 **To** 20 'Effectuer le code si N est dans la plage 8 à 20

End Select

* + 1. Le mot clé **Is** associé à un opérateur de comparaison (=, <>, <, <=, > ou >=) permet de spécifier une restriction sur les valeurs correspondantes de l'expression. Si le mot clé **Is** n'est pas indiqué, il est automatiquement inséré.

Select Case N

Case **Is** >= 5 'Effectuer le code si N supérieur ou égal à 5.

End Select

On peut utiliser plusieurs expressions ou plages dans chaque clause Case (séparées par des virgules).

Par exemple, la ligne suivante est valide :

Case 1 **To** 4, 7 **To** 9, 11, 13, **Is** > MaxNumber

On peut aussi indiquer des plages et des expressions multiples pour des chaînes de caractères.

Dans l'exemple suivant, Case correspond aux chaînes qui sont absolument identiques à «aaa», aux chaînes comprises entre «ccc» et «ddd» dans l'ordre alphabétique, ainsi qu'à la valeur de Var :

Case "aaa", "ccc" To "ddd", Var

Pour les 'Pro':

Les "Case" peuvent contenir n'importe quelle expression. Aussi il est possible de tester dans les conditions, non pas les valeurs d'une même variable, mais divers fonctions totalement indépendantes ou comme ici des fonctions travaillant toutes sur une même variable. C'est un usage méconnu du **Select Case** qui clarifie l'écriture et qui évite de multiples If Then ou Goto.

1. **Je retiens l’essentiel.**

|  |
| --- |
| * La structure de contrôle **Select Case** permet de créer une structure décisionnelle permettant d'exécuter un grand nombre de blocs de code différents en fonction de la valeur d'une expression.   **Select Case** expression  **Case** valeur1 code effectué si expression=valeur1  **Case** valeur2 code effectué si expression=valeur2  **Case** valeur3 code effectué si expression=valeur3  …  **Case** Else code effectué dans tous les autres cas  End Select   * Dans la structure de contrôle **Select Case**, plusieurs clauses d'expression peuvent être séparées par des virgules. * Le mot clé To permet de définir les limites d'une plage de valeurs correspondantes pour N. * Le mot clé **Is** associé à un opérateur de comparaison (=, <>, <, <=, > ou >=) permet de spécifier une restriction sur les valeurs correspondantes de l'expression. Si le mot clé **Is** n'est pas indiqué, il est automatiquement inséré. |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
2. Ecrire une syntaxe de la structure de contrôle select case.
3. Expliquez le mot clé To associé dans une structure de contrôle select case.

**Rep1 :** la syntaxe de la structure de contrôle select case

Select Case expression

Case valeur1 'code effectué si expression=valeur1

Case valeur2 'code effectué si expression=valeur2

Case valeur3 'code effectué si expression=valeur3…

Case Else 'code effectué dans tous les autres cas

End Select

**Rep2 :** dans la boucle select case, le mot clé **To** permet de définir les limites d'une plage de valeurs correspondantes pour N.

* 1. **En utilisant mes connaissances**

Comment spécifie la dernière expression de la boucle Select Case ?

Rep : pour spécifier la dernière expression de la boucle Select Case nous utilisons le mot clé **Case Else.**

* 1. **Pour aller plus loin**

Ecrire un programme en MS VB 2010 express permettant d’afficher si l’apprenant est débutant (1), ancien (2) ou finaliste (3) après avoir entré le numéro (1 ou 2 ou 3) de sa classe au clavier.

**Leçon 12 : Utilisation de la structure de contrôle du Do Loop.**

1. **Je saurai:**

* Définir la structure Do Loop ;
* Distinguer les mots clé de la structure de contrôle Do Loop ;
* Utiliser la structure de contrôle Do Loop.

1. **J’observe et je réfléchi:**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Image 12.1 Fenêtre montrant la structure de contrôle du « Do Loop»** | | |

**Activité 1 : Observation**

Je découvre que la boucle Do Loop peut être utilisée avec condition **While** ou **Until**

**Activité 2 : Définir la structure de contrôle du Do Loop**

Do Loop est une construction d'itération (instruction de contrôle) de script qui exécute une ou plusieurs instructions pendant ou jusqu'à ce qu'une condition logique soit remplie

**Activité 3 : Distinguer les mots clé de la structure de contrôle Do Loop**

Cette structure est utilisée dans un programme, lorsqu’il est nécessaire de répéter une ou plusieurs actions jusqu'à ce que quelque chose se passe, ou tant que quelque chose ne s'est pas passé.

**La structure de contrôle « Do Loop » utilise les différents mots clés à savoir :**

Je donne la fonction de chaque mot clé de la structure de contrôle « Do Loop »

**Loop** = Boucle (qui permet de faire une boucle, recommencer la boucle);  
**Do** = Faire (pour effectuer, faire les actions qui suivent);  
**While** = Tant que (permettant de réaliser une condition, Ex :Tant que X = 1, ou Tant que le mot de passe est erroné);  
**Until** = Jusqu'à ce que (lui aussi permet de réaliser une condition Jusqu'à ce que X = 10, ou Jusqu'à ce que le mot de passe soit correct).

**Activité 4 Utiliser la structure de contrôle Do Loop**

J’utilise la structure de contrôle Do Loop :

La boucle **Do Loop** permet de faire des boucles mais sans que le nombre de boucle (d'itération) soit déterminé au départ.

Comme **Do … Loop** est une boucle sans fin, (sans arrêt)Il faut une condition d'arrêt qui détermine la sortie de la boucle qui est Until (Jusqu'à ce que) ou While (Tant que) avant la condition d'arrêt pour sortir de la boucle.

**Je donne la syntaxe la structure de contrôle Do Loop.**

**Do Until** (condition)

Code

**Loop**

Boucler jusqu'à ce que condition soit vraie.

Cela signifie, Si la condition est fausse, effectuer le code, boucler et recommencer le code jusqu'à ce que condition soit égale à True.

Retenons qu’avant de débuter la boucle, la condition doit être fausse sinon la boucle ne sera jamais exécutée et qu’à chaque boucle la condition est évaluée.

**Exercice 1 :** Ecrire un programme en MS VB 2010 Express qui permet d’afficher un nombre négatif, il affiche **« entrer un nombre négatif »** jusqu’à ce que la condition soit réalisé.

Public Class Form1

Dim R As Integer

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

Do

R = InputBox("Entrer un nombre négatif")

Loop While R >= 0

End Sub

End Class

**Exercice 2** **:** Ecrire un programme qui permet d’afficher les entiers de 1 à 10, il quitter si le conteur atteint à 11

Public Class Form1

Dim i As Integer = 0

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Button2.Text = "LOOP UNTIL"

Do

i = i + 1

MsgBox(i.ToString)

Loop Until i = 10

Exit Sub

End Sub

**Exercice 3** **:** Ecrire un programme en MS VB 2010 Express qui permet d’entrer un nombre supérieur à 10, il affiche **« donnez un nombre <10 »** jusqu’à c’est que la condition soit réaliser.

Public Class Form1

Dim M, K As Integer

Private Sub Button3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click

Button3.Text = "While"

While M < 10

M += 1

M = InputBox("entrer un nombre < à 10")

End While

End Sub

End Class

1. **Je retiens l’essentiel.**

|  |
| --- |
| Do Loop est une construction d'itération (instruction de contrôle) de script qui exécute une ou plusieurs instructions pendant ou jusqu'à ce qu'une condition logique soit remplie  La fonction de chaque mot clé de la structure de contrôle « Do Loop » est :  **Loop** = Boucle (qui permet de faire une boucle, recommencer la boucle);  **Do** = Faire (pour effectuer, faire les actions qui suivent);  **While** = Tant que (permettant de réaliser une condition, Ex : Tant que X = 1, ou Tant que le mot de passe est erroné);  **Until** = Jusqu'à ce que (lui aussi permet de réaliser une condition Jusqu'à ce que X = 10, ou Jusqu'à ce que le mot de passe soit correct).  **Les différentes syntaxes de la boucle Do Loop est :**  **1er forme :**  Une boucle Do While...Loop tourne tant que sa condition d'itération est vraie (comme Do Until...Loop) et elle teste en amont de chaque boucle.  Do While «condition »    Instruction Loop  **2e forme :**  Une boucle Do...Loop While tourne tant que sa condition d'itération est vraie, cette dernière est testée à chaque fin de boucle, donc le programme y passe forcément au moins une fois.  Do     Instruction Loop While Condition  **3e forme :**  Do Until...Loop est une boucle qui itère tant que sa condition de sortie est fausse (comme Do Until...Loop), cette dernière est testée à chaque début de boucle, donc le programme n'y passe pas forcément  Do Until Condition  Instruction Loop  **4e Do...Loop Until**  Do...Loop Until est une boucle qui itère tant que sa condition de sortie est fausse, cette dernière est testée à chaque fin de boucle, donc le programme y passe forcément au moins une fois.  **Do**  instruction  **Loop Until** Condition |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
      * 1. Donnez la syntaxe da la boucle Do Loop … Until
        2. Quand est ce qu’on utilisé la structure de contrôle Do Loop

Rep1. Do

**Instruction**

Loop Until **(condition)**

Rep2.

* 1. **En utilisant mes connaissances**

1. Y a-t-il autre moyen de remplacer la structure de contrôle Do Loop par une autre boucle ? Laquelle ?
2. Ecrire un programme permettant d’entrer le nombre « **5** » et afficher « **le nombre incorrect**» dans le cas contraire.

Rép1 :

Rép2 :

* 1. **Pour aller plus loin**

**Distinguez les arguments rencontrés dans la boucle Do … Loop**

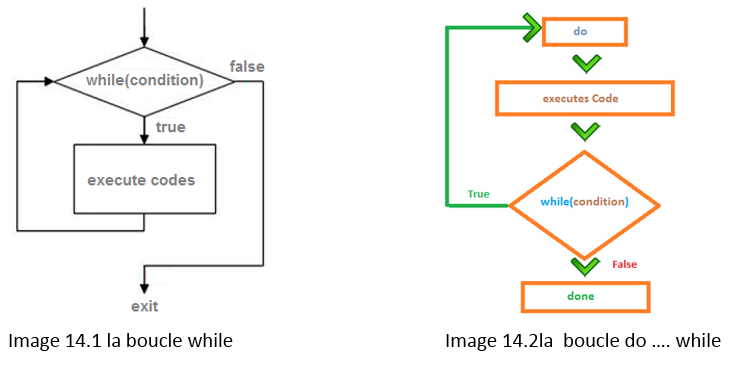
| **Argument** | **Description** |
| --- | --- |
| condition | Expression logique dont l'évaluation a pour résultat True ou False. |
| statements | Tout groupe d'une ou plusieurs instructions de script QlikView. |
| while / until | Les clauses conditionnelles while ou until ne doivent figurer qu'une fois dans une instruction do…loop, soit après do, soit après loop. Chaque condition n'est interprétée que la première fois, mais elle est évaluée à chaque fois que le script la rencontre dans la boucle. |
| exit do | Si une clause exit do se trouve dans la boucle, l'exécution du script est transférée à la première instruction qui suit la clause loop indiquant la fin de la boucle. Il est possible de rendre une clause exit do conditionnelle par l'utilisation facultative d'un suffixe when ou unless. |

**Leçon 13 : Utilisation de la structure de contrôle While … end while.**

1. **Je saurai:**

* Définir la structure **While … end While** ;
* Distinguer les mots clé associés à la boucle **While … end While** ;
* Utiliser la structure de contrôle **While … end While**.

1. **J’observe et je réfléchi:**



**Activité 1 : Observation**

Je constate que la boucle while répète l’instruction si la réponse de la condition est vrais et quitter dans le cas contraire.

**Activité 2 : Définir la structure de contrôle While**  **… end While**

While est une structure de contrôle (boucle) qui tourne autant de fois tant qu'une condition est vraie. Cette boucle se traduit en français en : « Tant que Condition ... Fin Tant que ».

**Activité 3 : Distinguer les mots clé de la structure While**  **… end While**

Dans la boucle while, on rencontre le mot « **Exit While** » pour sortir de la boucle; il existe aussi le mot «**Continue While**» qui permet de sauter au prochain End While et de poursuivre la boucle.

**Activité 4 : utilisation de la boucle While**  **… end While**

|  |  |
| --- | --- |
| **While** Condition  Instruction à traiter  **End While** |  |

Exemple: on incrémente un compteur, on sort quand il est égal à 20.

Dim Counter As Integer = 0

While Counter < 20 ' Test la valeur du compteur.

Counter += 1 ' Incrémente le compteur.

End While

1. **Je retiens l’essentiel.**

|  |
| --- |
| While est une structure de contrôle (boucle) qui tourne autant de fois tant qu'une condition est vraie.  **While** Condition  Instruction à traiter  **End While**  Dans la boucle while, on rencontre le mot « **Exit While** » pour sortir de la boucle; il existe aussi le mot «**Continue While**» qui permet de sauter au prochain End While et de poursuivre la boucle. |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**

Pourquoi dites –vous que While est une boucle ?

Reponse

La structure de contrôle While est une boucle parce qu’elle repend l’instruction autant de fois jusqu’à c’est que la condition soit réalisée.

* 1. **En utilisant mes connaissances**

A l’aide d’un schéma, montre le fonctionnement d’une structure de contrôle **While**.

Reponse

|  |  |
| --- | --- |
|  | * L’instruction d’entrée qui peut être initiation ou autres instructions * Condition While qui admet une réponse soit vraie ou faux. * Si la réponse de la condition est vraie, on exécute l’instruction qui suit la condition et on retourne à la condition. * Si la réponse de la condition est fausse on quitte la boucle While |

* 1. **Pour aller plus loin**

Ecrire un programme en MS VB 2010 Express permettant d’entre un nombre entre 17 et 25.Il affiche « plus grand » si le nombre est supérieur à 25 ou « plus petit » dans le cas contraire.

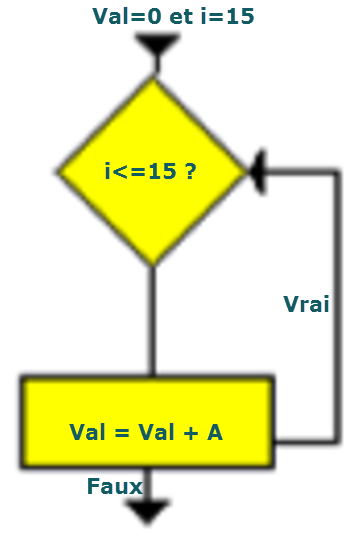
**Réponse :**

**Leçon 14 : Utilisation de la structure de contrôle For … Each.**

1. **Je saurai:**

* Donner la structure et la syntaxe de la boucle **For Each** ;
* Utiliser la boucle **For Each**.

1. **J’observe et je réfléchi:**



**Activité 1 : Observation**

Je découvre que **For Each** est un boucle qui retourne le résultat de la condition est vrai et quitte si le résultat de la condition est faux.

**Activité 2 : Donner la structure de la boucle For Each**

**For Each** est une variante de la boucle **For** … mais elle permet de parcourir les objets d'une collection. Elle n'utilise pas l'indice.

Prenons par exemple, un contrôle ListBox il a une collection items qui contient tous les éléments de la ListBox

ListBox.item(0) contient la première ligne

ListBox.item(1) contient la seconde ligne

ListBox.item(2) contient la troisième.

**Activité 3 : Donner les différentes syntaxes de la boucle For Each**

|  |  |
| --- | --- |
| **For Each** Valiable **in** **ListBox.** Valiable  Expression  **Next** | Public Class Form2  Private Sub Form2\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  Dim listVal As String  Dim item As Object  For Each item In ListBox1.Items  listVal = item  MsgBox(listVal)  Next  End Sub  End Class |
| **For Each** Valiable **As Objet in ListBox.** Valiable  Expression  **Next** | Public Class Form1  Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  Dim mystring As String  Dim item As Object  For Each item In ListBox1.Items  mystring = item  Next  End Sub  End Class |
| **For Each** Valiable As String In chaine  If Condition Then  Instruction  End If  **Next** | Public Class Form1  Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  Dim mystring As String  For Each item As Object In ListBox1.Items  mystring = mystring + item  Next  End Sub  End Class |

**Attention**, dans une boucle **For** **Each**, on peut parcourir la collection mais on ne peut pas modifier un élément de la collection.

La variable de boucle peut être déclarée après **For** comme dans l’exemple suivant:

**Dim** mystring **As** String

**For Each** item **As** Objet **in** ListBox.items

mystring=mystring+item

**Next**

Donc, au lieu de déclarer Item comme un objet, on aurait pu le déclarer comme un **ListBox.Item**. Cette notion de collection est beaucoup plus large.

Dans l’exemple suivant, nous allons tester chaque caractère dans une String, et voir s'il est égal à "i", en utilisant For Each:

**Dim** chaine **As** String = "aeiou"

**Dim** c **As** String

**For Each** car **As** String In chaine

**If** car= "**i**" **Then**…

**Next**

Retenons que dans une boucle **For Each**, on peut parcourir la collection mais on ne peut pas modifier un élément de la collection.

**Activité 4: Utiliser la boucle For … Each.**

J’utilise la boucle For … Each.

Ecrire un programme en MS VB 2010 Express permettant d’afficher les provinces de notre pays.

Je lance une application, j’insère un form et un List Box dans lequel je saisie les provinces de notre pays enfin j’entre les codes suivants.

Public Class Form1

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

Dim mystring As String

Dim item As Object

For Each item In ListBox1.Items

mystring = item

MsgBox(mystring)

Next

End Sub

Private Sub ListBox1\_SelectedIndexChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles ListBox1.SelectedIndexChanged

End Sub

End Class

**Exemple 1 :** écrire un programme en MS VB 2010 express permettant d’entrer un nombre positif no nul.

**Exemple 2 : En utilisant la boucle while,** Ecrire un programme en MS VB 2010 Express qui permet d’afficher la somme et la moyenne de cinq notes entrés au clavier.

1. **Je retiens l’essentiel.**

|  |
| --- |
| **For Each** est un boucle qui retourne le résultat de la condition est vrai et quitte si le résultat de la condition est faux.  **For Each** est une variante de la boucle **For** … mais elle permet de parcourir les objets d'une collection. Elle n'utilise pas l'indice.  Retenons que dans une boucle **For Each**, on peut parcourir la collection mais on ne peut pas modifier un élément de la collection.  La syntaxe de la boucle For Each  For Each Varable In Liste des éléments  Instruction  Next |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**

Enumérez les mots clés rencontrés dans la structure de contrôle For Each.

* 1. **En utilisant mes connaissances**

Ecrire un programme en MS VB 2010 Express qui permet de parcourir une chaine de caractère en comptant et en affichant la redondance de la voyelle « a » après il affiche le nombre d’occurrence de cette voyelle.

* 1. **Pour aller plus loin**

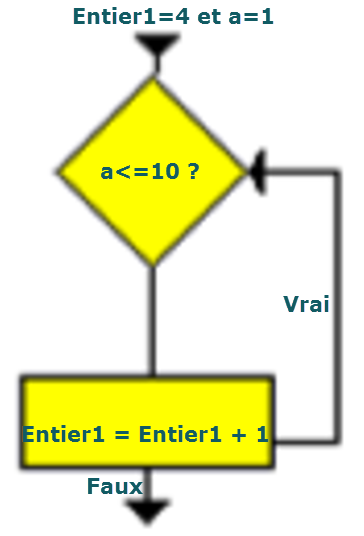
*Ecrire un programme en MS VB 2010 Express qui permet de parcourir une chaine de caractère fournit par l’utilisateur en comptant et en affichant la redondance de la 1er voyelle après il affiche le nombre d’occurrence de cette voyelle.*

**Leçon 15 : Utilisation de la structure de contrôle For … Next.**

1. **Je saurai:**

* Donner la structure et la syntaxe de la boucle **For Next**;
* Utiliser la boucle **For … Next**.

1. **J’observe et je réfléchi:**



**Image 17.1 montre la structure de fonctionnement de la boucle for … next.**

**Activité 1 : Observation**

Je découvre que la boucle **For… Next** répète l’instruction proposé si la condition est vraie et quitte si la condition est fausse ou bien s’il rencontre l’expression **Exit For**.

**Activité 2 : Donner la description de la boucle For … Next**

**Je décris la boucle For … Next**

Une boucle **For** s'exécute un certain nombre de fois, relativement prédéfinie car la valeur du compteur peut changer en cours de route.

Le **For** … **Next** permet de faire les boucles. Sachant que les boucles sont très utilisées pour parcourir une plage de valeur, par exemple : « parcourir tous les éléments d'un tableau ou pour effectuer de manière itérative un calcul ».

Avec **For … Next**, le nombre de boucle va être déterminé par une variable qui sert de compteur: la variable de boucle.

Le nombre d'exécution est déterminé au départ de la boucle car le compteur a une valeur de départ, une valeur d'arrêt.

Pour variable allant de 'début' à 'fin' : **For i=10 To 3**

La boucle **For…Next** est composée forcement au minimum par trois mots clé : For, To, Next et deux variable, l’une pour initialisation l’autre pour le compteur.

For a = 1 To 10

Entier1 = Entier1 + 1

Next

Quelquesfoison utilise le mot clé Step pour spécifie le saut du compteur et Exit For pour sortir de la boucle sans attendre la fin du compteur.

Public Class Form1

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

Dim m As Double = 0.5

Dim K As Integer

For K = 1 To 10

m = m + m

If m = 2 Then

Exit For ‘cette phrase permet d’arrêter la boucle prématurément juste

à la 2e exécution de la boucle.

End If

Next K

MsgBox(K)

MsgBox(m)

End Sub

End Class

**Activité 3 : Donner la syntaxe de la boucle For … Next.**

**Je donne la syntaxe de la boucle For … Next**

For Initialisation To derniere valeur du compteur Step valeur de pas

Instructions du programme

Next

**Activité 4: Utiliser la boucle For … Next.**

Je donne quelques exemples d’utilisation de la boucle **For … Next**

Ecrire un programme qui permet d’afficher la somme des nombres impaire inférieur à 10

Public Class Form1

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

Dim Entier1 As Integer = 0

For a = 1 To 10 Step 2

Entier1 = Entier1 + a

Next

MsgBox(Entier1) ' dans ce cas la machine affiche le nombre 25

End Sub

End Class

1. **Je retiens l’essentiel.**

|  |
| --- |
| * la boucle **For… Next** répète l’instruction proposé si la condition est vraie et quitte si la condition est fausse ou bien s’il rencontre l’expression **Exit For**. * Elle permet de faire les boucles répétitives. Sachant que les boucles sont très utilisées pour parcourir une plage de valeur, par exemple : « parcourir tous les éléments d'un tableau ou pour effectuer de manière itérative un calcul ». * Avec **For … Next**, le nombre de boucle va être déterminé par une variable qui sert de compteur: la variable de boucle. * Le nombre d'exécution est déterminé au départ de la boucle car le compteur a une valeur de départ, une valeur d'arrêt. * La boucle **For…Next** est composée forcement au minimum par trois mots clé : **For**, **To**, **Next** et deux variable, l’une pour initialisation l’autre pour le compteur.   For Initialisation To derniere valeur du compteur Step valeur de pas  Instructions du programme  Next |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
      1. Donnez les différents mots clés rencontre dans la boucle **For … Next**.
      2. Comment faire pour arrêter la boucle **For … Next** d’une façon prématuré ?
   2. **En utilisant mes connaissances**

Analysez attentivement le programme et répondre aux questions suivantes.

Public Class Form1

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

Dim compt As Integer = 4

For a = 1 To 15

compt = compt + 3

Next

MsgBox(compt)

End Sub

End Class

1. Le programme est-il exécutable ?
2. Quelle est le compteur de la boucle ?
3. Quelles sont les valeurs que le compteur va prendre ?
4. Quelle est la variable qui contient la valeur à afficher ?
5. La machine affichera combien à la sortie de la boucle ?
6. Le pas est combien dans ce programme ?

**Pour aller plus loin**

***Ecrire un programme permettant de demander le nombre des apprenant de votre classe, d’entrer leurs notes de l’interrogation de mathématique sur 20, afficher la sommes, la moyenne, le maximum et le minimum des notes.***

Ecrire un programme en MS VB 2010 Express qui permet d’entrer un nombre supérieur à 10, il affiche « donnez un nombre <10 » jusqu’à c’est que la condition soit réaliser.

Public Class Form1

Dim i As Integer = 0

Private Sub Button4\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click

Button4.Text = "For...Next"

For Me.i = 1 To 10

MsgBox(i.ToString)

Next i

End Sub

End Class

**Leçon 16 : Exercices d’auto-évaluations.**

Questions :

1. Créer une variable 'i' de type Integer, initialiser la avec la valeur 2, Incrémenter i (ajouter 1).
2. Créer une variable 's' de type virgule flottante simple précision, initialiser la avec la valeur 12,7561 , créer une variable 's1' qui devra contenir la partie entière de 's' créer une variable s2 qui devra contenir 's' arrondi à 2 décimales après la virgule (Comme pour les valeurs monétaires). Que contiendra s1 et s2 à la fin?
3. Créer un nombre virgule flottante double précision nommé 'x', l'initialiser avec 123456,45. Calculer le cube de x (x puissance 3), la racine 4ieme de x :
4. Créer une variable 'x' de type Integer; pour toute valeur de x, afficher dans un label 'Label1' "Le cube de 12 est 1728" (exemple si x=12).
5. Demander dans une InputBox à l'utilisateur de taper un nombre entier. Multiplier ce nombre par 2, afficher le résultat dans une MessageBox. (avec l'instruction VB MsgBox puis avec la Classe MesssageBox du Framework).
6. Même chose que l'exercice 5 mais afficher uniquement le résultat si l'utilisateur a bien tapé dans la InputBox une valeur numérique. Si la saisie n'est pas numérique une MessageBox doit indiquer 'Erreur de saisie'.
7. Même chose que l'exercice 5 mais en demandant de taper un nombre avec 2 chiffres après la virgule (Calcul sur des Single). Gérer le fait que l'utilisateur peut se tromper et ne pas employer le bon séparateur décimal (en France s'il tape un point au lieu de la virgule par exemple). Utiliser pour la conversion String vers Single une instruction de conversion spécifique (pas CType).
8. Écrire une boucle qui affiche les nombres pairs de 2 à 100 dans le label 'label1'. Affiche 2 puis 4, 6 ,8,....100.
9. Écrire une boucle qui affiche les nombres allant d'un nombre demandé à l'utilisateur et descendant de ce nombre jusqu'à 1 mais n'affichant pas le nombre 4 ( si l'utilisateur tape 8 cela affichera: 8 puis 7 ,6 ,5 ,3 ,2 ,1).
10. Afficher le plus grand nombre possible dont le carré est inférieur à 1000. En d'autres termes, écrire une boucle qui affiche dans label1 les nombres croissant 1,2 ,3, 4...tant ce que le nombre au carré est inférieur à 1000. Utiliser While pour cette boucle et une variable 'Counter'.
11. Chercher l'erreur dans ce code qui affiche dans une boite de message les résultats de la table de multiplication de 1 à 9:

Dim i, j as Integer

|  |  |
| --- | --- |
| Public Class Form1  Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  Dim i, j As Integer  For i = 1 To 9  For j = 1 To 9  MsgBox(i \* j.ToString)  Next i “erreur de fermeture”  Next j i “erreur de fermeture”  End Sub  End Class | Public Class Form1  Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  Dim i, j As Integer  For i = 1 To 9  For j = 1 To 9  MsgBox(i \* j.ToString)  Next j  Next i  End Sub  End Class |

1. Demander par une InputBox à l'utilisateur de taper un chiffre entre 1 et 12, vérifier que ce chiffre est bien compris entre 1 et 12 ; si ce n'est pas le cas, reposer la question (utiliser une boucle Do.. loop pour boucler en cas de mauvaise réponse.)

Si l'utilisateur tape sur 'Annuler' dans la InputBox cela retourne une chaîne vide et cela plante. Comment gérer cela?

1. Faire une boucle avec Do Loop, tournant de 1 à 100 et additionnant à une variable 'somme' à chaque tour la variable de boucle.

**Réponses:**

1. Créer une variable 'i' de type Integer, initialiser la avec la valeur 2, Incrémentez i (ajouter 1).

Public Class Form1

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

Dim i As Integer=2

'i = i + 1

i += 1

MessageBox.Show(i.ToString)

End Sub

End Class

1. Créer une variable 's' de type virgule flottante simple précision, initialiser la avec la valeur 12,7561 , créer une variable 's1' qui devra contenir la partie entière de 's' créer une variable s2 qui devra contenir 's' arrondi à 2 décimales après la virgule (Comme pour les valeurs monétaires). Que contiendra s1 et s2 à la fin?

Public Class Form1

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

Dim s As Single

Dim s1 As Single

Dim s2 As Single

s = 12.7561

s1 = Math.Truncate(s)

s2 = Math.Round(s, 2)

MessageBox.Show(s.ToString)

MessageBox.Show(s1.ToString)

MessageBox.Show(s2.ToString)

End Sub

End Class

s1 sera égal à 12. (partie entière) s2 sera égal à 12.76 (arrondi à l'entier le plus proche)

On remarque qu'on a déclaré toutes les variables au début (c'est une bonne manière de faire), plutôt qu'au moment ou on en a besoin.

On a bien écrit s =12.4561 et pas s =12,4561 car le séparateur décimal pour les littéraux est le point.

Au lieu d'écrire s1 =Math.Truncate(s), on écrit souvent s1 =Int(s) en utilisant une instruction Visual Basic.

Question : pourquoi Math. avant Truncate? C'est pour indiquer l'espace de nom System.Math qui contient Trunccate; on aurait pu aussi écrire en tête de module Imports System.Math et ensuite s1 =Truncate(s) aurait été accepté.

1. Créer un nombre virgule flottante double précision nommé 'x', l'initialiser avec 123456,45. Calculer et afficher dans une boite de dialogue : le cube de x (x puissance 3), la racine 4ieme de x :

Public Class Form1

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

Dim x As Double

Dim x1 As Double

Dim x2 As Double

x = 123456.45

x1 = Math.Pow(x, 3)

x2 = Math.Pow(x, 1 / 4)

MessageBox.Show(x.ToString)

MessageBox.Show(x1.ToString)

MessageBox.Show(x2.ToString)

End Sub

End Class

Prendre la racine Nieme d'un nombre revient à le mettre à la puissance 1/N. donc racine 4eme de X : x2= Math.Pow(X, 1/4)

1. Créer une variable 'x' de type Integer; pour toute valeur de x afficher dans un label 'Label1' "Le cube de 12 est 1728" (exemple si x=12).

Public Class Form1

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

Dim X As Integer

X = 5

Label1.Text = "Le cube de " & X.ToString & " est " & (Math.Pow(X, 3)).ToString

End Sub

End Class

1. Demander dans une InputBox à l'utilisateur de taper un nombre entier. Multiplier ce nombre par 2, afficher le résultat dans une MessageBox (avec l'instruction VB MsgBox puis avec la Classe MesssageBox du Framework).

Public Class Form1

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

Dim s As String

Dim i As Integer

s = InputBox("Test", "Taper un nombre entier")

i = CType(s, Integer)

i = i \* 2

MsgBox(i.ToString)

MessageBox.Show(i.ToString)

L'InputBox retourne une String, il faut la transformer en Integer, effectuer le calcul puis la retransformer en String pour l'afficher.

1. Même chose que l'exercice 4.2 mais afficher uniquement le résultat si l'utilisateur a bien tapé dans la InputBox une valeur numérique. Si la saisie n'est pas numérique une MessageBox doit indiquer 'Erreur de saisie'.

Public Class Form1

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

Dim s As String

Dim i As Integer

s = InputBox("Test", "Taper un nombre entier")

If IsNumeric(s) Then

i = CType(s, Integer)

i = i \* 2

MsgBox(i.ToString)

Else

MsgBox("Erreur de saisie")

End If

End Sub

End Class

Noter que le code entre If et Else et entre Else et End If est décalé à droite par l'ajout d'espace ou de Tab, ce qui permet une meilleure lecture du code.

1. *Même chose que l'exercice 4.2 mais en demandant de taper un nombre avec 2 chiffres après la virgule (Calcul sur des Single). Gérer le fait que l'utilisateur peut se tromper et ne pas employer le bon séparateur*

Public Class Form1

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

Dim i, j As Integer

For i = 1 To 9

For j = 1 To 9

MsgBox(i \* j.ToString)

Next j

Next i

End Sub

End Class

Comme le séparateur décimal, sur un ordinateur français (Culture Fr) est le ',', on remplace les points par des virgules avant de convertir en Single.

On remarque que le résultat est affiché avec une virgule car ToString utilise le séparateur de la culture en cours.

1. Écrire une boucle qui affiche les nombres pairs de 2 à 100 dans le label 'Label1'. Affiche 2 puis 4, 6 ,8,....100.

Public Class Form1

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

Dim i as Integer

For i = 2 To 100 Step 2

Label1.Text = i.ToString

Label1.Refresh()

Next i

End Sub

End Class

Ici, comme on connaît les valeurs de début et de fin, on utilise une boucle For Next.

Step permet de 'boucler' de 2 en 2.

Ne pas oublier Label1.Refresh qui force l'affichage pour chaque tour de la boucle. Sans cela l'affichage serait mis à jour uniquement en fin de procédure.

Noter que le code entre For et Next est décalé à droite par l'ajout d'espace ou de Tab, ce qui permet une meilleure lecture du code.

1. Écrire une boucle qui affiche les nombres allant d'un nombre demandé à l'utilisateur et descendant de ce nombre jusqu'à 1 mais n'affichant pas le nombre 4 ( si l'utilisateur tape 8 cela affichera: 8 puis 7 ,6 ,5 ,3 ,2 ,1).

Public Class Form1

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

Dim i As Integer

Dim sfin As String

Dim fin As Integer

sfin = InputBox("Donner un nombre entier")

fin = CInt(sfin) 'on transforme sfin, une String saisie par l'utilisateur en fin , un Integer.

For i = fin To 1 Step -1

If i <> 4 Then

Label1.Text = i.ToString

Label1.Refresh()

End If

Next i

End Sub

End Class

Ici on ne connaît pas la valeur de la fin de la boucle mais elle sera saisie par l'utilisateur et on peut la mettre dans la variable nommée 'fin'. La boucle c'est une boucle descendante, il faut donc un pas négatif: Step -1. On pourrait aussi vérifier par un If fin>1 Then que la valeur de fin n'est pas inférieur ou égale à 1.

1. Afficher le plus grand nombre possible dont le carré est inférieur à 1000. En d'autres termes, écrire une boucle qui affiche dans label1 les nombres croissant 1,2 ,3, 4...tant ce que le nombre au carré est inférieur à 1000. Utiliser While pour cette boucle et une variable 'Counter'.

A priori, on ne connaît pas la valeur de fin de boucle, donc on va utiliser une boucle While plutôt que For Next.

Il faut gérer soit même la variable de boucle et l'incrémenter mais en fin de boucle.

Public Class Form1

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

Dim Counter As Integer = 0

While Counter \* Counter < 1000 ' Test la valeur du compteur.

Label1.Text = Counter.ToString

Counter += 1 ' Incrémente le compteur.

End While

End Sub

End Class

1. Chercher l'erreur dans ce code qui affiche dans une boite de message les résultats de la table de multiplication de 1 à 9:

|  |  |
| --- | --- |
| Public Class Form1  Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  Dim i, j As Integer  For i = 1 To 9  For j = 1 To 9  MsgBox(i \* j.ToString)  Next i  Next j  End Sub  End Class | Public Class Form1  Private Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load  Dim i, j As Integer  For i = 1 To 9  For j = 1 To 9  MsgBox(i \* j.ToString)  Next j  Next i  End Sub  End Class |

Réponse: erreur sur les variables dans les 2 Next: si le premier For utilise la variable de boucle i, c'est le dernier Next qui doit indiquer la variable i: La boucle interne doit tourner DANS la boucle externe;

1. Demander par une InputBox à l'utilisateur de taper un chiffre entre 1 et 12, vérifier que ce chiffre est bien compris entre 1 et 12 ; si ce n'est pas le cas, reposer la question (utiliser une boucle Do.. loop pour boucler en cas de mauvaise réponse.):

Public Class Form1

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

Dim rep As String

Dim r As Integer

Do

rep = InputBox("Tapez un chiffre entre 1 et 12")

r = CType(rep, Integer)

Loop Until r > 0 And r < 13

MsgBox(rep)

End Sub

End Class

Si l'utilisateur tape sur 'Annuler' dans la InputBox cela retourne une chaîne vide et cela plante. Comment gérer cela?

Public Class Form1

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

Dim rep As String

Dim r As Integer

Do

rep = InputBox("Tapez un chiffre entre 1 et 12")

If rep = "" Then

Exit Do

End If

r = CType(rep, Integer)

Loop Until r > 0 And r < 13

MsgBox(rep)

End Sub

End Class

Exit Do permet de sortir de la boucle Do Loop.

1. Faire une boucle avec Do Loop, tournant de 1 à 100 et additionnant à une variable 'somme'à chaque tour la variable de boucle.

Public Class Form1

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

Dim i As Integer = 0

Dim somme As Integer

Do

i = i + 1

somme = somme + i

Loop Until i = 100

MsgBox(somme.ToString)

End Sub

End Class

[**https://fr.wikibooks.org/wiki/Programmation\_Visual\_Basic\_.NET/Boucles#Boucle\_For\_Each**](https://fr.wikibooks.org/wiki/Programmation_Visual_Basic_.NET/Boucles#Boucle_For_Each)

[**http://gaetan.dussaux.free.fr/cours/java/7.htm**](http://gaetan.dussaux.free.fr/cours/java/7.htm)

[**https://fr.wikipedia.org/wiki/Structure\_de\_contr%C3%B4le**](https://fr.wikipedia.org/wiki/Structure_de_contr%C3%B4le)

[**https://openclassrooms.com/fr/courses/1894236-programmez-avec-le-langage-c/1895634-les-structures-de-controle**](https://openclassrooms.com/fr/courses/1894236-programmez-avec-le-langage-c/1895634-les-structures-de-controle)

[**http://www.info-3000.com/vbvba/boucledoloopwhileuntil.php**](http://www.info-3000.com/vbvba/boucledoloopwhileuntil.php)

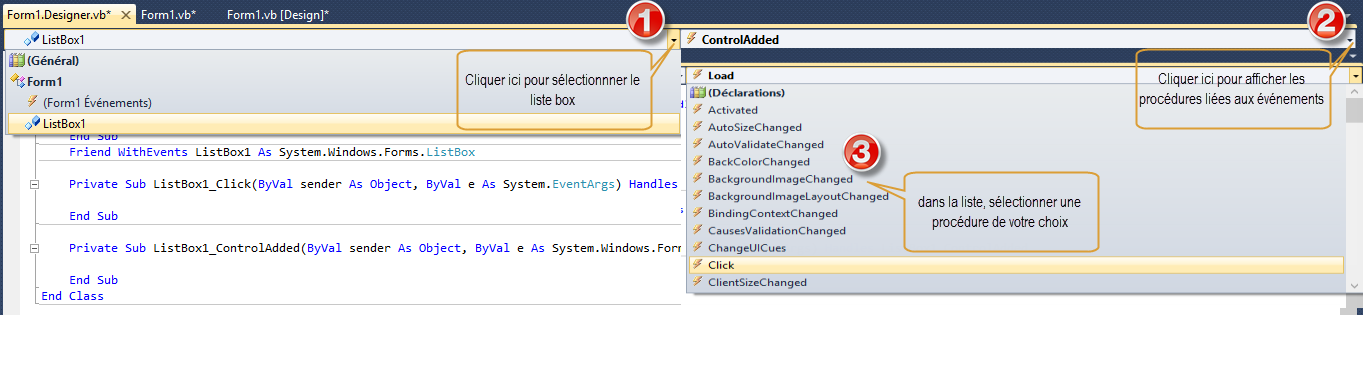
**Objectif pédagogique 3.Utiliser les procédures et les paramètres**

**Leçon 17 : Distinction des procédures**

1. **Je saurai:**

* Définir une procédure;
* Distinguer les différentes sortes de procédures ;
* Utiliser les procédures.

1. **J’observe et je réfléchi:**



**Image 17.1 montre les contrôles d’une procédure.**

**Activité 1 : Définition d’une procédure**

Je définis une procédure

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

Une procédure est généralement commencer par Sub et se terminer par End Sub (ou Function…End Function).

**Activité 2 : Distinction des différentes sortes de procédures.**

En VB nous distinguons deux sortes des procédures qui sont :

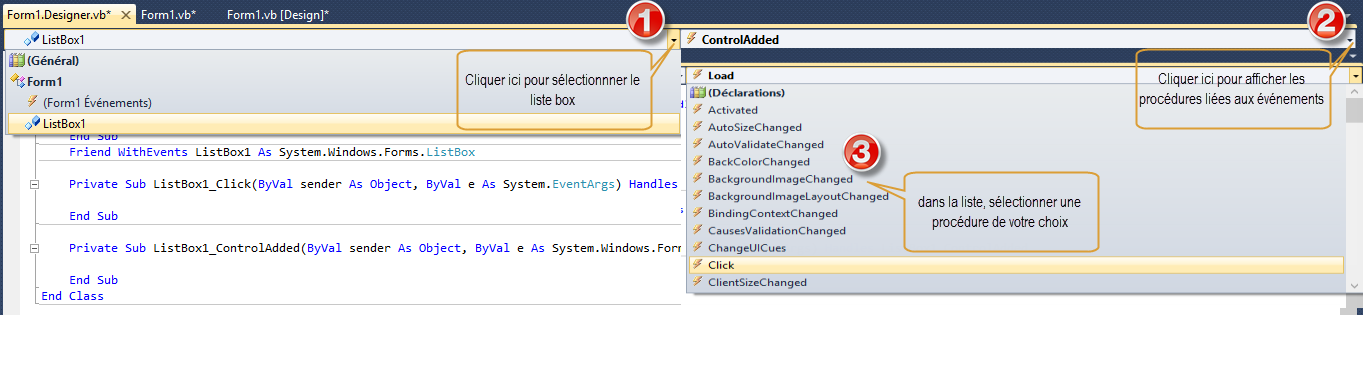
* **Les procédures Sub:** Elles débutent par le mot Sub et se terminent par End Sub.
* **Les procédures Function:** Elles débutent par Function et se terminent par End Function.

**Je différencie les deux types des procédures en VB :**

En VB, il y a deux types de procédures: les procédures liées aux évènements et celles qui ne sont pas liées.

Pour ajouter une procédure liée à une fenêtre, je procède comme suit :

* Je clique sur l’outil **dérouler** pour sélectionner la **listebox (1)**,
* Je clique sur l’outil **dérouler** pour afficher les différentes procédures possible **(2)**,
* Je sélectionne la procédure à utiliser **(3)**,
* enfin j’entrer dans la procédure pour le programme.

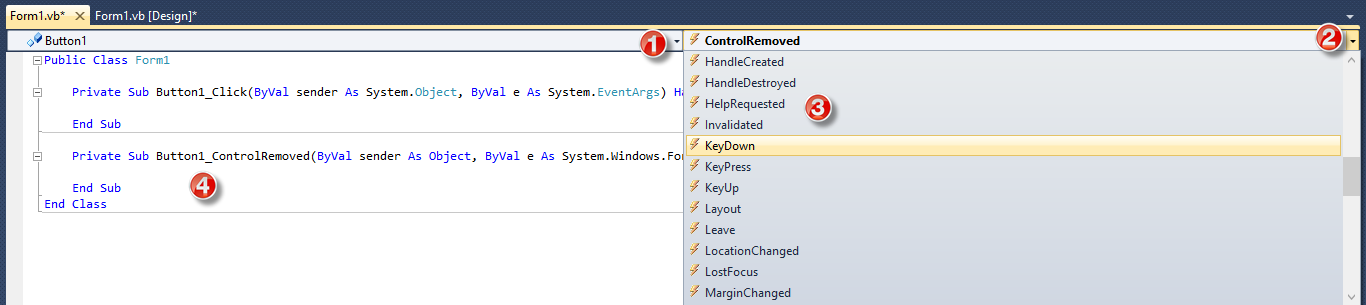


Pour voir les **procédures lies** à une fenêtre (Form) on fait un double sur cette fenêtre (Form).

La procédure liée au bouton est celle qui contient le code à effectuer quand l'utilisateur clique sur le bouton.

Pour ajouter une procédure liée au bouton, je procède comme suit :

* Je fais un double clic sur un bouton à programmer,
* Je clique sur l’outil **dérouler** pour sélectionner le **bouton (1)**,
* Je clique sur l’outil **dérouler** pour afficher les différentes procédures possible **(2)**,
* Je sélectionne la procédure à utiliser **(3)**, enfin j’entrer dans la procédure pour le programme donc c'est là que l'on écrit le code qui doit s'effectuer lorsque l'utilisateur clique sur le bouton.



Pour voir les **procédures lies** au bouton, on fait un double sur ce bouton dans un Form.

**Procédures non liées**

Procédures non liées sont des codes qui font des tâches particulières, qui sont utilisés à plusieurs endroits et qui ne sont pas liées à des événements.

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.

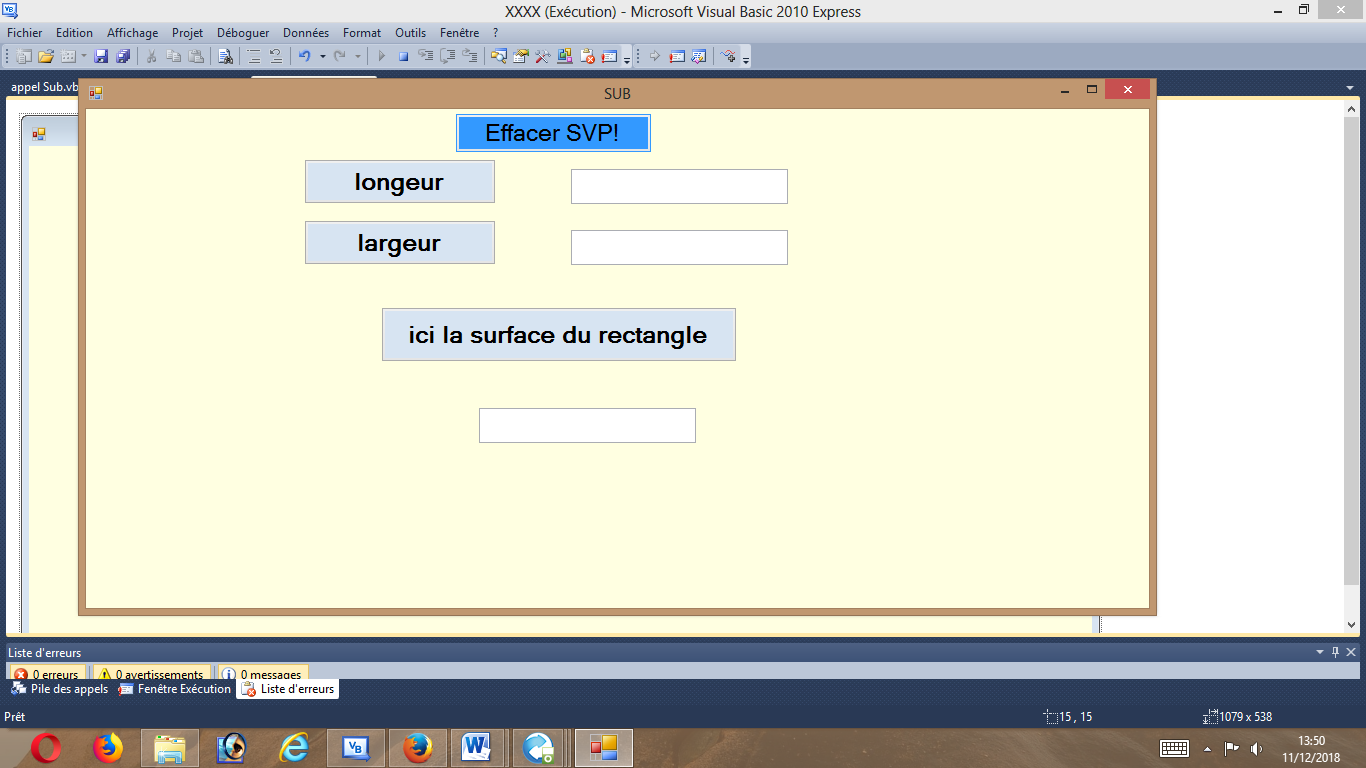
Pour appeler une procédure nous utilisons la syntaxe suivant : **Call** NomdeProcedure()

**Activité 3 : Utilisation de procédure Sub.**

A l’aide d’un exemple, j’explique l’utilisation de procédure Sub:

**Exemple:** Ecrire en programme dans visual basic 2010 ‘’en utilisant Sub ‘’ qui permet de calculer la surface du rectangle.

La résolution de l’exercice est la suivante :



Public Class Form1

Dim longueur, largeur As Double

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

Button1.Text = "longueur"

Button1.Text = TextBox1.Text

End Sub

Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Button2.Text = "largeur"

Button2.Text = TextBox2.Text

End Sub

Sub RESULTAT(ByVal Textbox1, ByVal Textbox2)

TextBox3.Text = Val(Textbox1.Text) + (Textbox2.Text)

End Sub

Sub TextBox3\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox3.TextChanged

TextBox3.Text = TextBox3.Text

End Sub

Sub TextBox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged

TextBox1.Text = TextBox1.Text

If Not IsNumeric(TextBox1.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Sub TextBox2\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox2.TextChanged

TextBox2.Text = TextBox2.Text

If Not IsNumeric(TextBox2.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

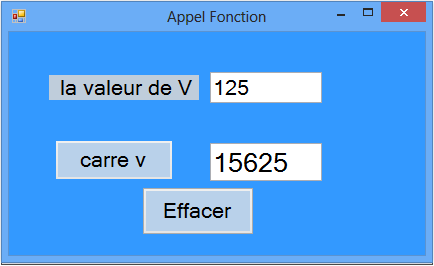
**Activité 4 : Utilisation de procédure Function.**

A l’aide d’un exemple, j’explique l’utilisation de procédure Function:

**Exemple** : Ecrire un programme qui affiche le carre d’une valeur **v** entrée au clavier ; ce carre est calculé par une fonction « **car** ».

**La résolution est la suivante :**

Création de la fenêtre suivante et écriture des codes suivants



Public Class Form1

Dim v As Single

Function Car(ByVal v As Single) As Single

Return v \* v

End Function

Private Sub TextBox2\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox2.TextChanged

TextBox2.Text = TextBox2.Text

If Not IsNumeric(TextBox2.Text) Then

MsgBox("Taper le numeric SVP!")

End If

End Sub

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

Button1.Text = "carre v"

v = TextBox2.Text

TextBox1.Text = Car(v)

End Sub

Private Sub TextBox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged

TextBox1.Text = TextBox1.Text

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Button2.Text = "Effacer"

TextBox1.Text = ""

TextBox2.Text = ""

End Sub

End Class

Ecrire un programme qui affiche le carre d’une valeur v entrée au clavier ; ce carre est calculé par une fonction « **car** ».

1. **Je retiens l’essentiel.**

|  |
| --- |
| * Une procédure est un ensemble d'instructions, de lignes de code, un groupement d'instructions bien définies effectuant une tache précise. * En VB, il existe généralement deux sortes de procédures. * **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.   Pour ajouter une procédure liée à une fenêtre, on procède comme suit :   * cliquer sur l’outil **dérouler** pour sélectionner la **listebox (1)**, * cliquer sur l’outil **dérouler** pour afficher les différentes procédures possible **(2)**,   sélectionner la procédure à utiliser **(3)**, enfin j’entrer dans la procédure pour la programme.  Pour ajouter une procédure liée au bouton, je procède comme suit :   * On fait un double clic sur un bouton à programmer, * cliquer sur l’outil **dérouler** pour sélectionner le **bouton (1)**, * cliquer sur l’outil **dérouler** pour afficher les différentes procédures possible **(2)**, * sélectionner la procédure à utiliser **(3)**, enfin entrer dans la procédure pour le programme donc c'est là que l'on écrit le code qui doit s'effectuer lorsque l'utilisateur clique sur le bouton. * En VB, il y a deux types de procédures: les procédures liées aux évènements et celles qui ne sont pas liées * Les procédures liées aux boutons sont celles qui contiennent les codes à effectuer quand l'utilisateur clique sur le bouton. * Procédures non liées sont des codes qui font des tâches particulières, qui sont utilisés à plusieurs endroits et qui ne sont pas liées à des événements. * Pour appeler une procédure nous utilisons la syntaxe suivant : **Call** NomdeProcedure() |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**

Distinguer les différentes sortes des procédures.

* 1. **En utilisant mes connaissances**

Distinguer les types des procédures.

1. **Pour aller plus loin**

Donnez une syntaxe pour appelerune fonction « **car** » qui calculele carre d’une valeur ; de l’exercice précédent.

**Leçon 18 : Distinction des paramètres**

1. **Je saurai:**

* Définir un paramètre;
* Reconnaitre la syntaxe d’un paramètre ;
* Utiliser un paramètre en VB.

1. **J’observe et je réfléchi:**

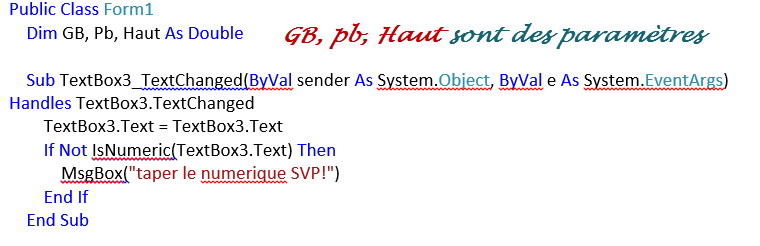


Image 18.1 montre la déclaration des paramètres «GB, Pb et Haut » sous MS VB 2010 Express.

**Activité 1 : Définition d’un paramètre**

Les paramètres sont des variables ou valeurs qui seront transmises à la procédure.

**Activité 2 : Reconnaitre la syntaxe d’un paramètre**

**Je décris la syntaxe de déclaration d’un paramètre sous MS VB 2010 Express.**

*Cas du procedure Sub :*

*Sub* ***le nom du procedure*** *(****paramètre1*** *As* ***type de parametre1****,* ***paramètre2*** *As* ***type de parametre2****)*

**Exemple*:***

*Sub proc (par1 As integer, par2 As integer)*

*…*

*End Sub*

*Cas du procedure Function :*

*Function* ***le nom d’une fonction*** *(paramètre1 As type de parametre1, paramètre2 As type de parametre2) As type résultat de la fonction*

**Exemple**:

Function Minimum(x As Integer, y As Integer) As Integer

If x<y Then

Return x

Else

Return y

End If

End Function

Exemple suivant est une procédure **SUB** nommée **MaProcédure** et recevant deux paramètres *(paramètre1, paramètre2)*.

**Activité 3 : Utilisation d’un paramètre**

*Sub MaProcédure (paramètre1, paramètre2)*

*...*

*End Sub*

Lors d'appel de la procédure, on envoie les paramètres:

Syntaxe d’appel : ***Call MaProcédure(2,3)*** signifie que la procédure est fournis deux paramètres qui sont 2 et 3.

**Notons que :**

* Il est nécessaire de définir le type des paramètres comme suis:

Sub **MaProcédure** (**paramètre1** As **Integer**, **paramètre2** As **Integer**) indique que la procédure attend deux paramètres de type entiers.

* Il y a deux manières d'envoyer des paramètres :

1° **Par valeur** : (By Val) c'est la valeur, le contenu de la variable qui est envoyé.

2°**Par référence** :(By Ref) c'est l'adresse (le lieu physique où se trouve la variable) qui est envoyée. Si la Sub modifie la variable, cette modification sera visible dans la procédure appelante après le retour.

* Si on a besoin que la procédure appelée retourne une valeur dans ce cas il faut créer une fonction.

**Exemple d’application :**

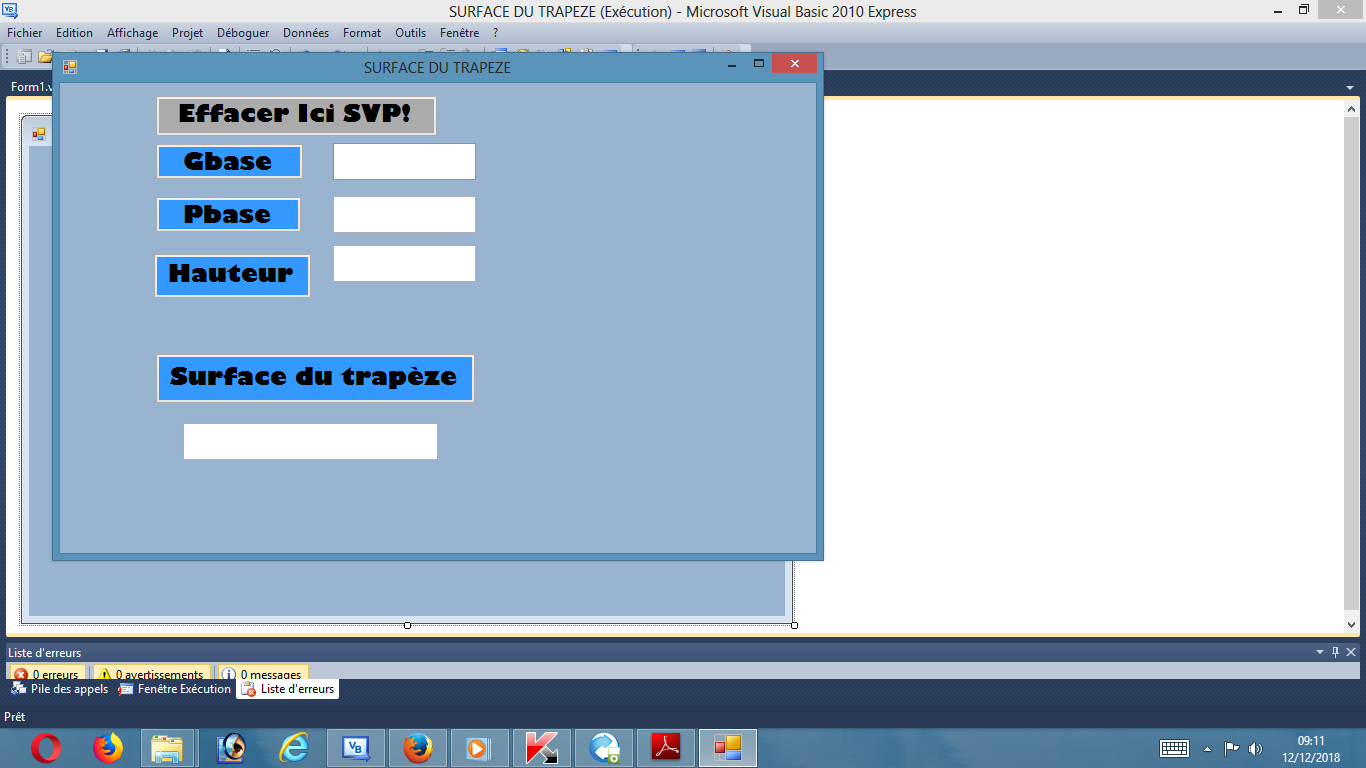
1. **Je retiens l’essentiel.**

|  |
| --- |
| * Les paramètres sont des variables ou valeurs qui seront transmises à la procédure. * les **syntaxes d’un paramètre dans une procédure**   *Cas d’une procédure Sub :*   * *Sub* ***le nom d’une procédure*** *(****paramètre1*** *As* ***type de parametre1****,* ***paramètre2*** *As* ***type de parametre2****)*   *Cas d’une procédure Function :*   * *Function* ***le nom d’une fonction*** *(paramètre1 As type de parametre1, paramètre2 As type de parametre2) As type résultat de la fonction*   Il y a deux manières d'envoyer des paramètres :  1° **Par valeur** : (By Val) c'est la valeur, le contenu de la variable qui est envoyé.  2°**Par référence** : (By Réf) c'est l'adresse (le lieu physique où se trouve la variable) qui est envoyée. Si la Sub modifie la variable, cette modification sera visible dans la procédure appelante après le retour. |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
      1. Définir un paramètre
      2. Donnez une syntaxe d’utiliser une un paramètre dans une procédure function.
   2. **En utilisant mes connaissances**

Ecrire un programme permettant de calculer et afficher la surface d’un trapèze des dimensions fournis au clavier en utilisant trois paramètres dans une procédure Sub.

Réponse :



Public Class Form1

Dim GB, Pb, Haut As Double

Sub TextBox3\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox3.TextChanged

TextBox3.Text = TextBox3.Text

If Not IsNumeric(TextBox3.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Sub TextBox2\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox2.TextChanged

TextBox2.Text = TextBox2.Text

If Not IsNumeric(TextBox2.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Sub TextBox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged

TextBox1.Text = TextBox1.Text

If Not IsNumeric(TextBox1.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Sub TextBox4\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox4.TextChanged

TextBox4.Text = TextBox4.Text

If Not IsNumeric(TextBox4.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Sub Button4\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles

Button4.Click

Button4.Text = "Surface du trapèze"

GB = TextBox1.Text

Pb = TextBox2.Text

Haut = TextBox3.Text

TextBox4.Text = Val((GB + Pb) \* Haut) / 2

End Sub

Sub Button5\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click

Button5.Text = "Effacer Ici SVP!"

TextBox1.Text = ""

TextBox2.Text = ""

TextBox3.Text = ""

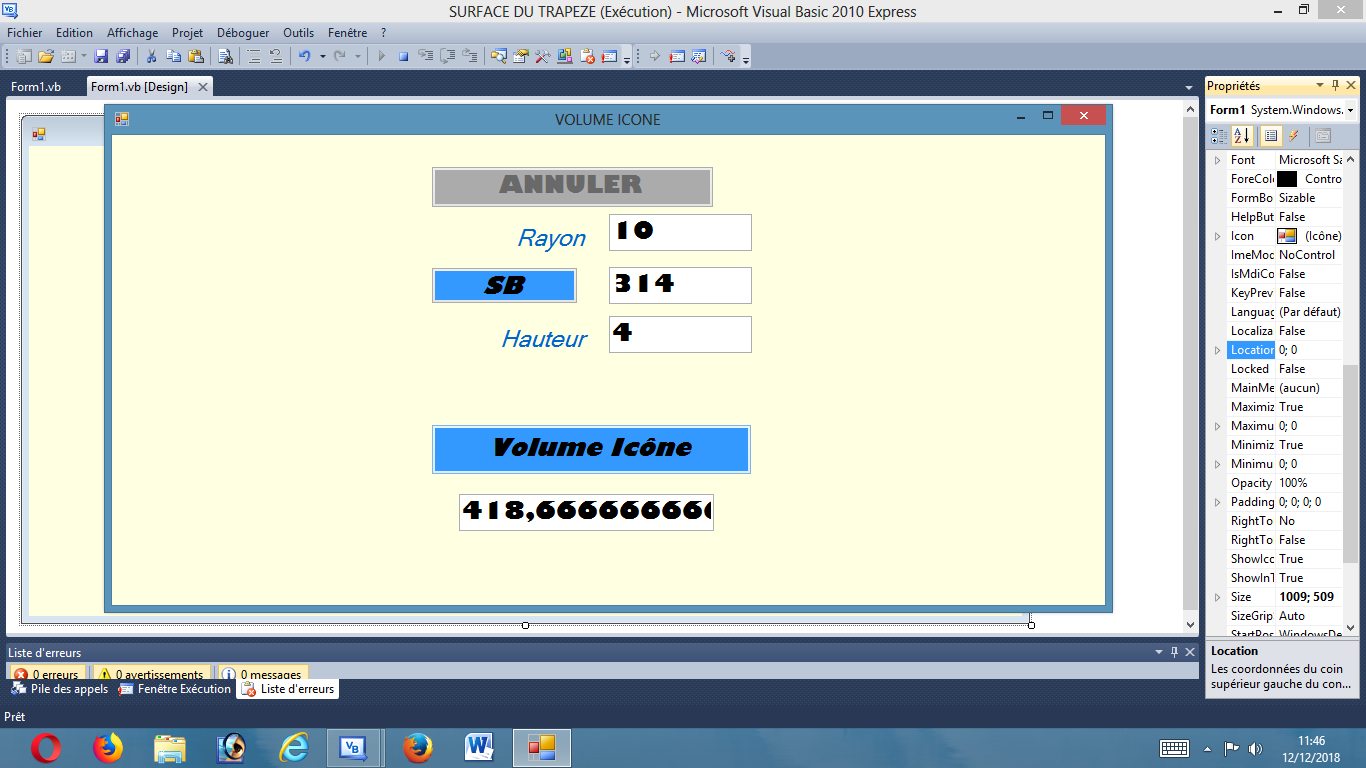
TextBox4.Text = ""

End Sub

End Class

**Pour aller plus loin**

Ecrire un programme permettant de calculer et afficher le volume d’un cône de dimensions fournis par l’utilisateur en utilisant trois paramètres dans une procédure Function.



Public Class Form1

Dim rayon, SBase, Haut As Double

Function VOLUME(ByVal SBase As Double, ByVal Haut As Double)

Return 1 / 3 \* SBase \* Haut

End Function

Sub TextBox3\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox3.TextChanged

TextBox3.Text = TextBox3.Text

If Not IsNumeric(TextBox3.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Sub TextBox2\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox2.TextChanged

TextBox2.Text = TextBox2.Text

If Not IsNumeric(TextBox2.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Sub TextBox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged

TextBox1.Text = TextBox1.Text

If Not IsNumeric(TextBox1.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Sub Form1\_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

End Sub

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

End Sub

Sub TextBox4\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox4.TextChanged

TextBox4.Text = TextBox4.Text

If Not IsNumeric(TextBox4.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Sub Button4\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click

Button4.Text = "Volume Icône"

Haut = TextBox3.Text

SBase = TextBox2.Text

TextBox4.Text = VOLUME(SBase, Haut)

End Sub

Sub Button5\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click

Button5.Text = "Effacer Ici SVP!"

TextBox1.Text = ""

TextBox2.Text = ""

TextBox3.Text = ""

TextBox4.Text = ""

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Button2.Text = "SB"

rayon = TextBox1.Text

TextBox2.Text = Val(rayon \* rayon \* 3.14)

End Sub

Private Sub Label1\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label1.Click

End Sub

End Class

**Leçon 19 : Différents types de classe sous MS VB 2010 Express**

1. **Je saurai:**

* Définir une classe;
* Décrire une classe ;
* Utiliser une classe.

1. **J’observe et je réfléchi:**

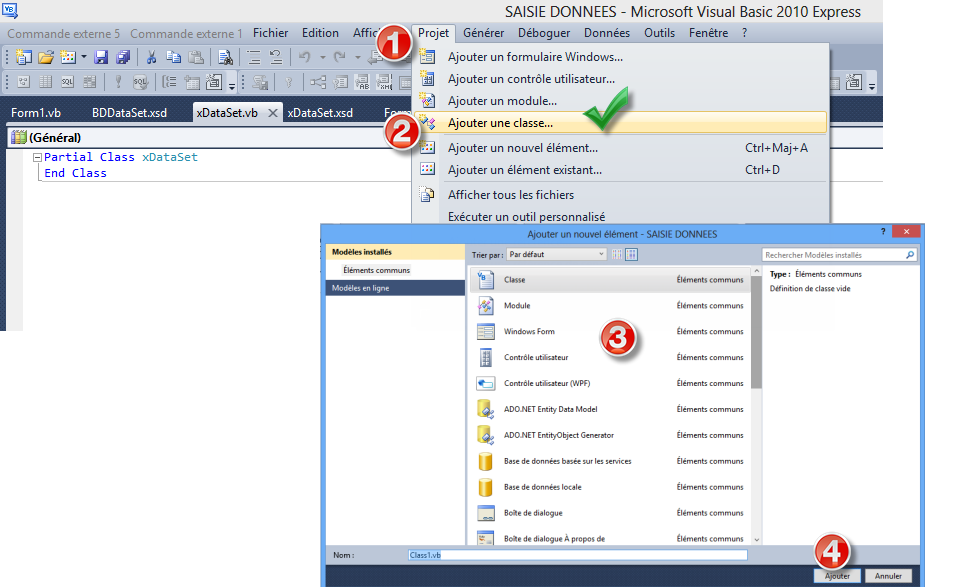


Image 19.1 montre les différentes étapes pour ajouter une classe ou autres éléments dans un projet MS VB 2010 Express.

Je découvre qu’on peut ajouter les différents éléments dans un projet.

**Activité 1 : Définition d’une classe**

Une classe est un type d’objet définissant les caractéristiques communes des autres objets (attributs et des méthodes).

**Activité 2 : Description d’une classe.**

La notion de classe est présente dans VB, elle se met en œuvre à travers une entité spéciale dénommée : le module de classe. Elle contient des attributs et des méthodes publiques ou privés, nous avons déclaré un module de classe : "Etudiant.cls" pour représenter la classe simple suivante où tout est publique :

Les classes sont regroupées en bibliothèques sous la dénomination Espace de noms (NameSpace).

Un Framework est un ensemble d'espace de noms et de classes.

Les Classes sont dans des espaces de nom qui sont dans des Dll (références). Ou Les ‘dll’ contiennent des **Espaces** **de** **noms** contenant des **Classes**.

Les références (dll) permettent de charger des composants, des Classes du Framework ou d'autres classes. L'instruction 'Imports' permet d'importer des espaces de nom venant de ses références. Cela donne accès dans le programme à des classes. On pourra donc instancier des objets grâce à ces classes puis utiliser des méthodes.

Noter que dans les Classes, il existe une structure arborescente comme suit:

* + La premier Classe (en haut) est System ;
  + Dessous il y a entre autres System.WindowsForms ;
  + Dessous System.Windows.Forms.Control;
  + Enfin System.Windows.Forms.Control.Name par exemple.

**Activité 3 : Distinction des types de classes.**

Il existe 3 types de Classes

1. *Les classes du Framework* fournies par Microsoft avec VB, Vista, Windows 7, Windows 8 et Windows 10. Elles sont communes à VB, C#, C et autres langages OO.

Il existe ainsi des classes :

• pour les formulaires Windows (WindowsForms) ;

• pour le Web (WebForms) ;

• pour l'accès aux données ;

• pour les réseaux ;

• pour la sécurité.

Notons qu’en créant un nouveau projet, les Classes le plus souvent utilisées sont automatiquement chargées dans ce projet.

1. *Les classes fournies par des tiers*

Ces sont des classe qu’on peut ajouter (des références (Dll)) permettant d'ajouter des nouvelles classes, cela permet d'ajouter de nouvelles fonctionnalités à VB.

**Exemple**: les pilotes de base de données, les contrôles qui n'existent pas dans VB et qu’on peut acheter.

1. *Les Classes écrites par le programmeur*

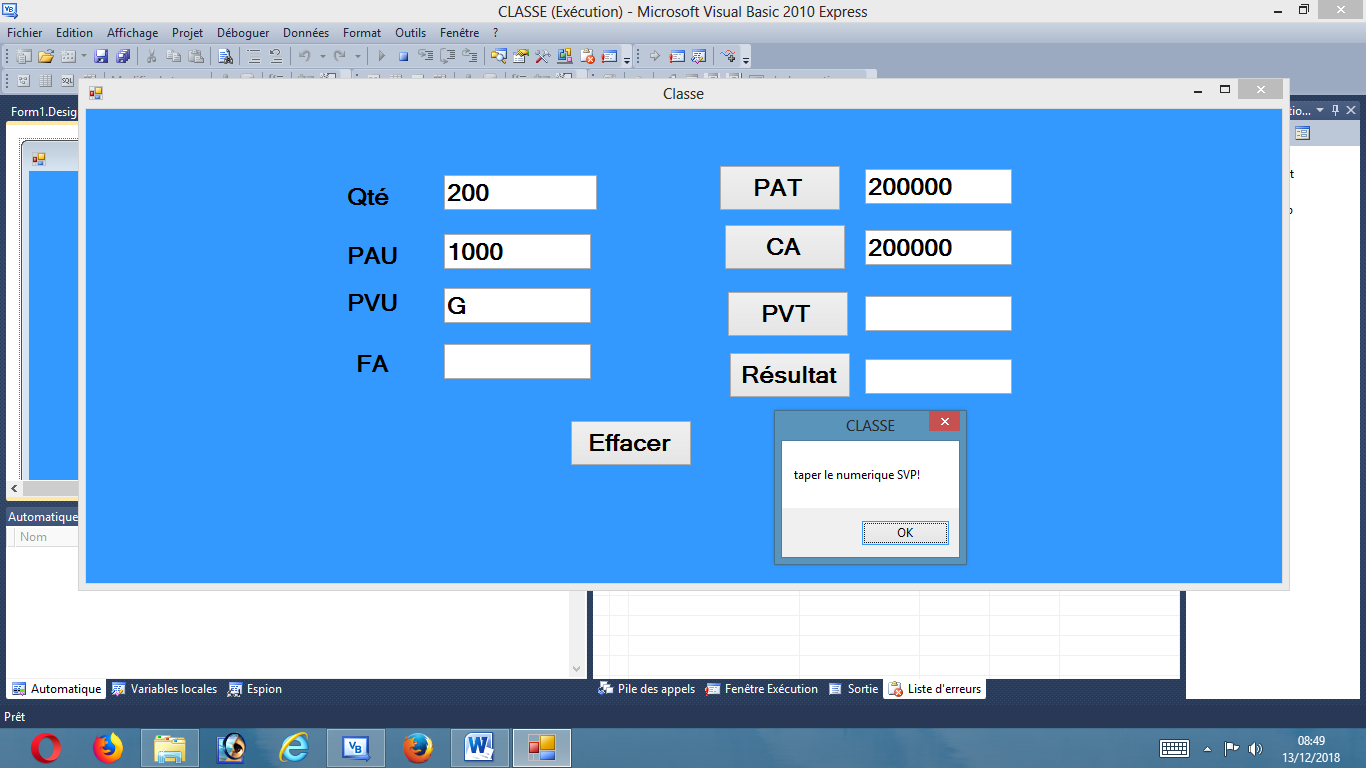
En VB, Le programmeur peut les créer de toute pièce dans les modules de Classe. Il peut créer une classe, ses propriétés, méthodes et l'utiliser dans le corps du programme.

On aussi créer une Classe, la compiler, puis dans un autre projet référencer la Dll de la classe déjà créée et l'utiliser dans le nouveau projet.

**Exemple d’application :**

Ecrire un programme qui calcule et affiche le résultat de commerce des marchandises dont la quantité  « en utilisant **Classe »**, le prix d’achat unitaire(PAU), le prix de vente unitaire(PVU) et Autres chargés consommés (FA) seront entrés au clavier pour que l’ordinateur fournisse le résultat.

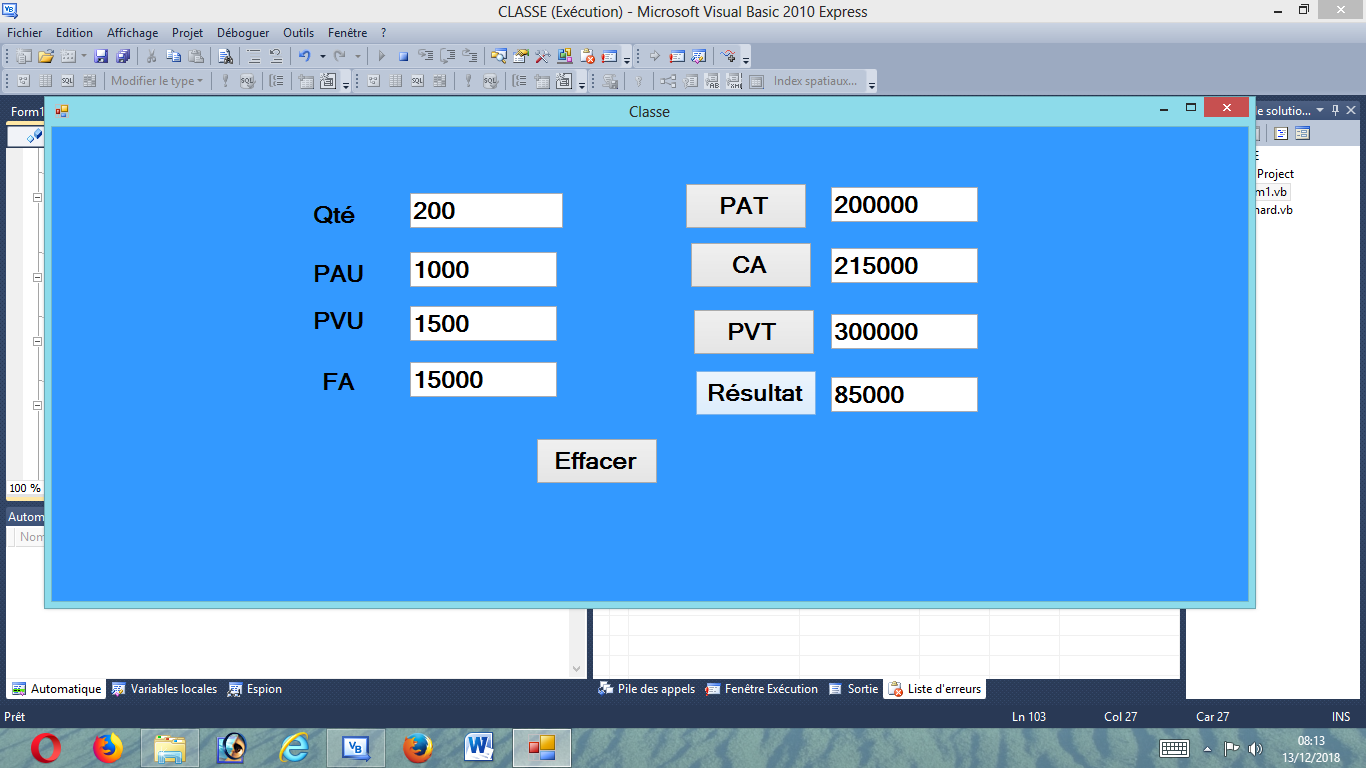
Résolution de l’exercice



L’application n’a pas exécuté car nous avons tapé une lettre **«  G »** au lieu d’un chiffre

Dans le champ **« PVU ».**

**Voilà l’application qui est correcte où nous allons trouver « le Résultat  de l’exercice»**



Public Class Form1

Dim Qt, PAU, PVU, FA As Double

Private Sub TextBox1\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox1.TextChanged

TextBox1.Text = TextBox1.Text

If Not IsNumeric(TextBox1.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Private Sub TextBox2\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox2.TextChanged

TextBox2.Text = TextBox2.Text

If Not IsNumeric(TextBox2.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Private Sub TextBox3\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox3.TextChanged

TextBox3.Text = TextBox3.Text

If Not IsNumeric(TextBox3.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Private Sub TextBox4\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox4.TextChanged

TextBox4.Text = TextBox4.Text

If Not IsNumeric(TextBox4.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Private Sub TextBox5\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox5.TextChanged

TextBox5.Text = TextBox5.Text

If Not IsNumeric(TextBox5.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Private Sub TextBox6\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox6.TextChanged

TextBox6.Text = TextBox6.Text

End Sub

Private Sub TextBox7\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox7.TextChanged

TextBox7.Text = TextBox7.Text

If Not IsNumeric(TextBox7.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

Private Sub TextBox8\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles TextBox8.TextChanged

TextBox8.Text = TextBox8.Text

If Not IsNumeric(TextBox8.Text) Then

MsgBox("taper le numerique SVP!")

End If

End Sub

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

Button1.Text = "PAT"

Qt = TextBox1.Text

PAU = TextBox2.Text

TextBox5.Text = Val(Qt \* PAU)

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

Button2.Text = "CA"

TextBox6.Text = Val(TextBox4.Text) + (TextBox5.Text)

End Sub

Private Sub Button3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click

Button3.Text = "PVT"

TextBox7.Text = Val(TextBox1.Text) \* (TextBox3.Text)

End Sub

Private Sub Button4\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click

Button4.Text = "Résultat"

TextBox8.Text = Val(TextBox7.Text) - (TextBox6.Text)

End Sub

Private Sub Label3\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Label3.Click

End Sub

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

End Sub

Private Sub Button5\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button5.Click

Button5.Text = "Effacer"

TextBox1.Text = ""

TextBox2.Text = ""

TextBox3.Text = ""

TextBox4.Text = ""

TextBox5.Text = ""

TextBox6.Text = ""

TextBox7.Text = ""

TextBox8.Text = ""

End Sub

End Class

CLASSE

Public Class Richard

Public Qt As Double

Public PAU As Double

Public PVU As Double

Public FA As Double

End Class

1. **Je retiens l’essentiel.**

|  |
| --- |
| * Une classe est un type d’objet définissant les caractéristiques communes des autres objets (attributs et des méthodes). * Les classes sont regroupées en bibliothèques sous la dénomination Espace de noms * (NameSpace). * Un Framework est un ensemble d'espace de noms et de classes. * Il existe 3 types de Classes qui sont : * *Les classes du Framework* fournies par Microsoft ; * *Les classes fournies par des tiers ;* * *Les Classes écrites par le programmeur*. |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
      1. Donnez les différents logiciels fournis avec *Framework*
      2. Donnez au moins trois langages de programmations partageant le framework.

**En utilisant mes connaissances**

Donnez une structure arborescente d’une classe

Rep :

Dans les Classes, il existe une structure arborescente comme suit:

* + La premier Classe (en haut) est System ;
  + Dessous il y a entre autres System.WindowsForms ;
  + Dessous System.Windows.Forms.Control;
  + Enfin System.Windows.Forms.Control.Name par exemple.

1. **Pour aller plus loin**

**Quels est la classe :**

1. **contenant l'accès aux bases de données sous VB.**
2. **contient la bibliothèque qui permet d'utiliser les instructions VB (MsgBox, IsNumeric, Chr, Asc.....).**

Rép :

**La classe contenant l'accès aux bases de données; ADO.Net en particulier est le System.data**

**La classe qui contient la bibliothèque qui permet d'utiliser les instructions VB (MsgBox, IsNumeric, Chr, Asc.....) est Microsoft.VisualBasic.**

**Leçon 20 : Ajout et suppression d’une Référence**

1. **Je saurai:**

* Ajouter une référence de bibliothèque d’objets ;
* Vérifier les références liées à un Project
* Supprimer une référence ;

1. **J’observe et je réfléchi:**

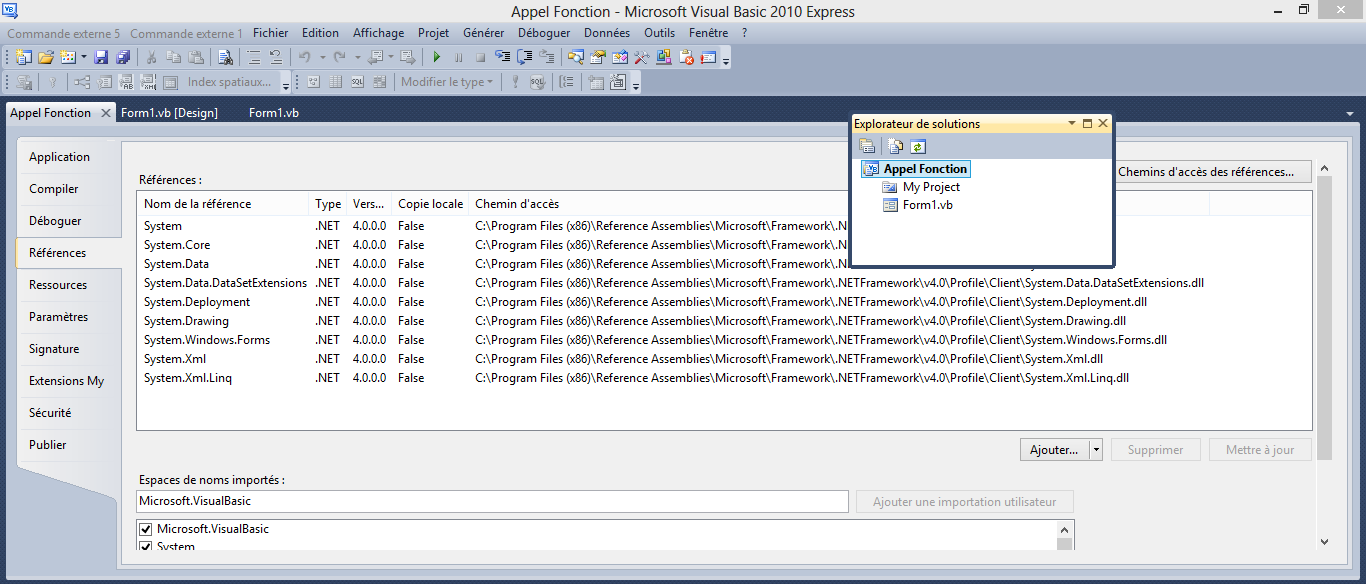


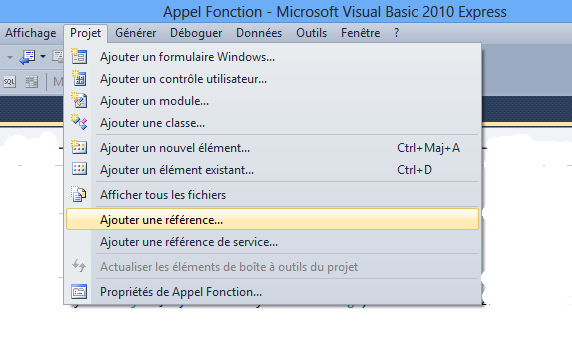
Image 20.1 montre l’appel d’une fonction en MS VB 2010 Express.

Je découvre qu’on peut ajouter, supprimer et mettre à jour la référence d’une fonction.

**Activité 1 : Insertion des Références**

**Je procède comme suis pour insérer une Références.**

Sous Microsoft Visual Basic 2010 Express, dans l’**Explorateur de solutions**, je fais un double-cliquez sur **My Projet**, ou dans le **menu Projet**, je clique sur **Références,** je clique sur **Ajouter** dans la **liste des** **Références** proposées sélectionner une **Références** enfin clique sur le menu **ok**.



On peut passer par l’onglet Projet, clic sur Ajouter une référence, sélectionner une référence enfin cliquer sur Ok

**Activité 2 : Vérification des références liées à un Project**

**Pour savoir si une application propose une bibliothèque d'objets, je procède comme suit :**

Sous Microsoft Visual Basic 2010 Express, dans l’**Explorateur de solutions**, je fais un double-cliquez sur **My Projet**, je clique sur **Références,** dans la **liste des** **Références** proposées je défile une liste de haut en bas pour visualiser toute la liste enfin je ferme la fenêtre.

**Activité 3 : Suppression d’une référence liée à un Project**

**Je procède comme suis pour supprimer une Références.**

Sous Microsoft Visual Basic 2010 Express, dans l’**Explorateur de solutions**, je fais un double-cliquez sur **My Projet**, ou dans le **menu Projet**, je clique sur **Références,** dans la **liste des** **Références** proposées, je sélectionne une **Références** à supprimer enfin je clique sur **Supprimer**.

1. **Je retiens l’essentiel.**

|  |
| --- |
| * La boîte de dialogue **références** affiche toutes les bibliothèques d’objets inscrits avec le système d’exploitation. * Dans l’Explorateur **d’objets**, vous pouvez sélectionner une classe dans la zone **Classes** et sélectionnez une méthode ou une propriété dans la zone **membres**. * Pour qu'une classe soit utilisée, il faut que le composant correspondant (la DLL) soit chargé; même si quelques composants du Framework (System.dll..) et le CLR (mscorlib.dll) étaient chargés, il faut vérifier et ajouter autres références aux bibliothèques des objets. |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
      1. Donnez le chemin le plus court pour insérer une référence sous MS VB 2010
      2. Quel est l’inconvénient de passer par ce chemin

Rep :

* + - 1. Clique sur l’onglet **Projet**, clic sur **Ajouter une référence**, sélectionner **une référence** souhaite et enfin cliquer sur **Ok**
      2. l’inconvénient de passer par ce chemin est qu’on n’est sûr qu’**une référence** est ajoutée car la liste n’est pas visualisée et pour ajouter plusieurs références on reprend toutes les étapes.
  1. **En utilisant mes connaissances**

En MS VB 2010, nous pouvons ajouter quels types de **référence**

Rep :

En MS VB 2010, nous pouvons ajouter deux types de référence : référence .net etréférence  COM.

**Pour aller plus loin**

Que faire si vous utilisez les objets d’autres applications dans le cadre de votre application Visual Basic et que vous souhaiterez peut-être établi une référence aux bibliothèques d’objets de ces applications ?

Rep : On doit d’abord être sûr que l’application fournit une bibliothèque d’objets après on ajouter ces références.

**Elément de la compétence 3 : Gérer les données**

**Objectifs pédagogiques 1 : Accéder aux données**

**Leçon 21 : Rappel sur la création et la gestion d’une base de données**

1. **Je saurai:**

* Définir la base de données ;
* Créer et connecter une base de données sous MS.VB 2010 Express ;
* Créer les tables d’une base de données

1. **J’observe et je réfléchi:**

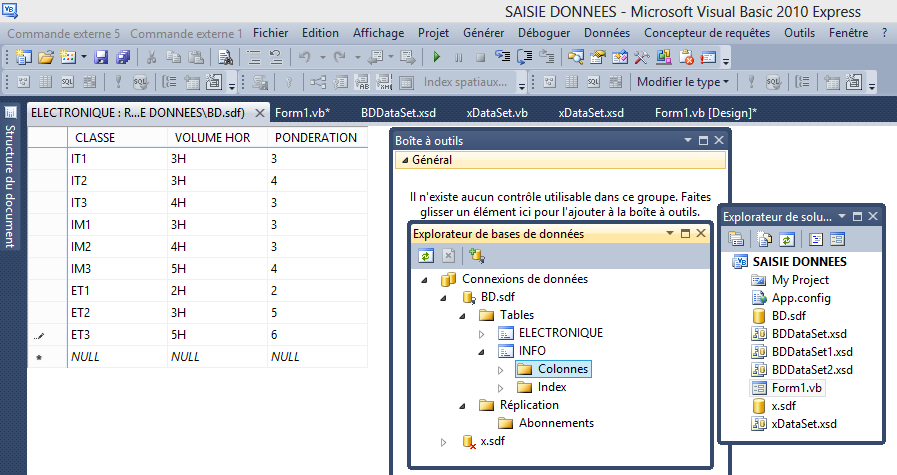


Image 22.1 Montrer les différentes fenêtres de création d’une base de données dans Microsoft Visual Basic 2010 Express

**Activité 1 : Observation**

Je découvre qu’en Microsoft Visual Basic 2010 Express, on peut créer une base de données et saisir les données.

**Activité 2 : Définition d’une base de données**

**Je définis une base de données comme étant une collection de données organisées de façon à être facilement accessible, administrées et mises à jour.**

**Activité 3 : Rappel sur la création d’une base de données Access 2013**

**Pour créer une base de données sous MS Access 2013, je procède comme suis :**

* Je lance le MS Access 2013 ;
* Je clique sur « base de données du bureau vide» ;
* Je saisie le nom de la base de données ;
* Le spécifie l’emplacement de ma base de données ;
* Je clique sur « créer » ;

**Je crée les tables dans ma base de données**

* Je clique sur l’onglet « Créer »
* Je clique sur l’outil « Table » du groupe « Tables »

**Je modifie les propriétés da la table créée :**

* Je fais un clic droit sur la table ;
* Je sélectionne « mode création »
* Je saisie le nom de la table ;
* Je clique sur OK;
* J’entre les propriétés de la table (nom du champ, type de données, description) ;

**Je spécifie la clé primaire de la table :**

* Je sélectionne le champ qui sera la clé primaire ;
* Je clique sur l’outil « clé primaire » de l’onglet « Création ».

**Je saisie les données dans une table :**

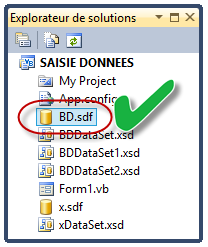
* J’ouvre la table en mode normal par double clics gauche de la souris ;
* J’entre les données dans les champs.

**Activité 3 : Création et connexion d’une base de données sous MS VB 2010 express**

Je donne les différentes étapes pour créer et connecter une base de données sous MS VB 2010 express.

* Lancer MS VB 2010 Express;
* Cliquer sur l’onglet « Données » ;
* Cliquer sur Ajouter une nouvelle source de données ;
* Cliquer sur « Base de données» ;
* Cliquer sur « Suivant» ;
* Cliquer sur « DataSet» ;
* Cliquer sur «Suivant» ;
* Cliquer sur « Nouvelle connexion» ;
* Entrer le nom de la base de données ;
* Cliquer sur Créer ;
* Cliquer sur «OK» ;
* Cliquer sur «Oui» ;
* Cliquer sur «Tester la connexion» ;
* Cliquer sur «Ok» ;
* Cliquer sur «Ok» ;
* Cliquer sur «Suivant» ;
* Cliquer sur «Oui» ;
* Cliquer sur «Suivant» ;
* Cliquer sur «Terminer».

Je clique sur l’onglet « Affichage» puis sur Explorateur de solution pour vérifier que ma base de données est bien créée et connectée. Voir la figuré ci contrée.



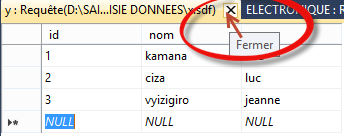
**Activité 4 : Décrire les étapes pour créer des tables, saisir, afficher et modifier des données dans des tables.**

**Je données les étapes pour créer une table**

* Dans l’explorateur de solution, je fais un double cliquer sur ma base de donnée ;
* Dans l’explorateur de base de données, je fais un clic droit sur « **Tables** » puis sur « **créer** **la** **table** »;
* J’entre le nom de la table, type de données, longueur, autorisation et je précise la clé primaire.
* Je clique sur « OK »

**Je données les étapes pour saisir les données dans la table**

* Dans l’explorateur de base de données, Je fais un clic bouton droit sur le nom de la table ;
* Je sélectionne « Afficher les données de la table »
* Je clic dans le champ de chaque colonne pour entrer les données ;
* Enfin, je clique sur « Fermer » voir figure ci-contre.



* Dans l’explorateur de base de données, Je fais un clic bouton droit sur le nom de la table ;

|  |  |
| --- | --- |
|  | **Dans l’explorateur de base de données, Je fais un clic droit sur la table (1) après :**   * Je peux déplacer la table (2) ; * Je peux consulter les propriétés des éléments de la table (3) ; * Je peux modifier les données dans la table (4) ; * Je peux créer et lancer des requêtes de la table (5) ; * Je peux afficher les enregistrements de la table (6) ; * Je peux copier la table vers un autre projet les enregistrements de la table (7) ; * Je peux consulter les propriétés de la table (8) |

**Je retiens l’essentiel.**

|  |
| --- |
| Pour créer une base de données sous MS Access 2013, je procède comme suis :   * Je lance le MS Access 2013 ; * Je clique sur « base de données du bureau vide» ; * Je saisie le nom de la base de données ; * Le spécifie l’emplacement de ma base de données ; * Je clique sur « créer » ;   **Sous MS VB 2010 Express, dans l’explorateur de base de données on peut :**   * Déplacer la table ; * Consulter les propriétés des éléments de la table; * Modifier les données dans la table; * Créer et lancer des requêtes de la table; * Afficher les enregistrements de la table; * Copier la table vers un autre projet les enregistrements de la table; * Consulter les propriétés de la table.   **Pour insérer la clé primaire de la table sous MS Access, on procède comme suit:**   * Sélectionner le champ qui sera la clé primaire ; * Cliquer sur l’outil « **Clé** **primaire** » de l’onglet « **Création** ». |

1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**
2. **Réponse 30**
   1. **En utilisant mes connaissances**

Comment saisir les données dans la table créée sous MS VB 2010 Express ?

**Pour aller plus loin**

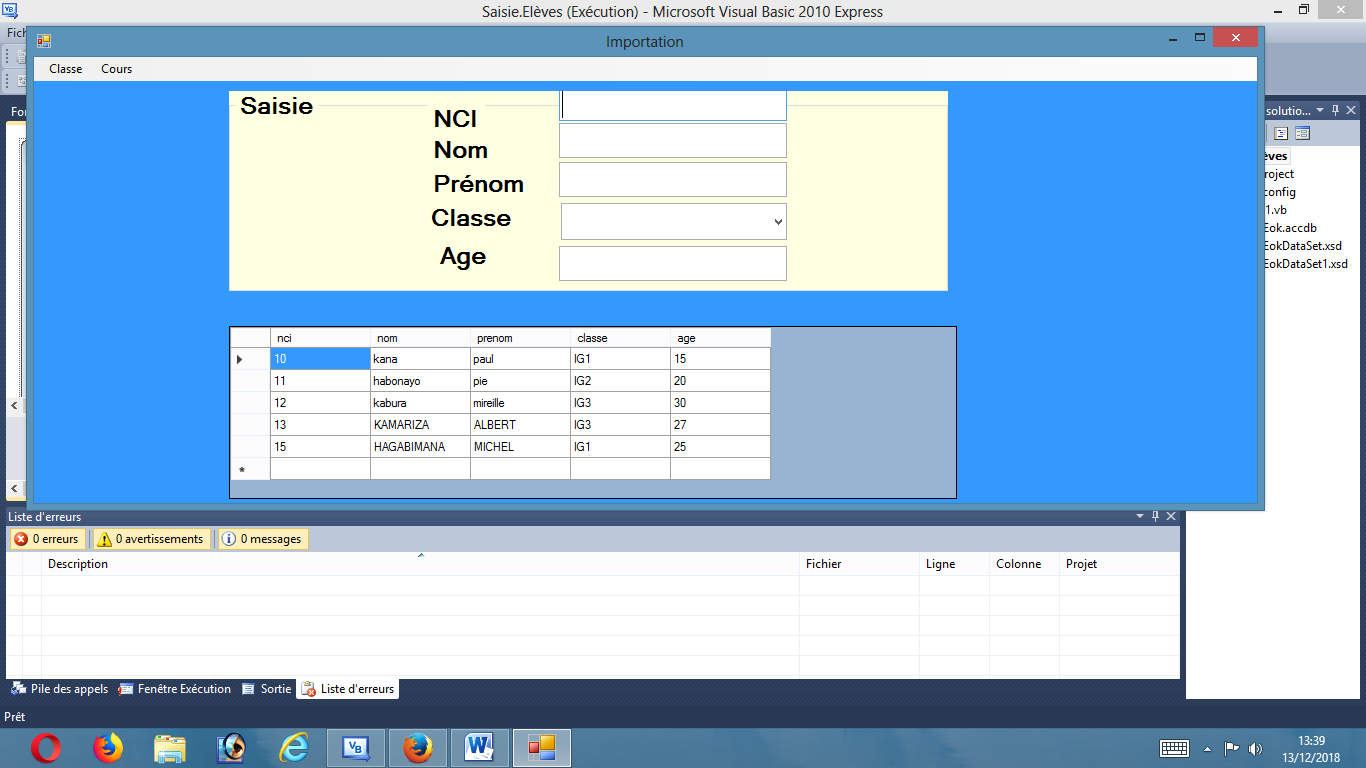
**Objectifs pédagogiques 2 : Manipuler les fichiers**

**Leçon 22 : Connection d’une application VB à une BD**

1. **Je saurai:**

* Définir la liaison de données ou Binding ;
* Donner les étapes d’importation des données ;
* Donner la méthode à utiliser pour le stockage des données dans l’Access 2010.

1. **J’observe et je réfléchi:**



**Image 21.1 Montrer comment importer les données de l’Access vers VB2010 express**

**Activité 1 : Observation**

Je découvre que les données qui se trouvent dans une base de données Access ont étés importées dans une fenêtre VB 2010 express.

**Activité 3 : définir la liaison de données ou Binding**

Les liaisons de données sont des processus qui établissent une connexion entre l'interface utilisateur de l'application et le logique métier. En d'autres termes, elles permettent d'établir une connexion entre un contrôle et une source de données. Cela permet d'afficher automatiquement le contenu d'une base de données ou d'une collection dans une DataGrid, une ListBox et autres éléments d’un formulaire.

Pour faire binding, il faut donc un objet visuel, la cible, (ListBox, TextBox) ayant une propriété de dépendante et faire une liaison avec la source de liaison qui est la propriété d'un objet (collection, tableau, base de données) La liaison peut être uni (OnWay= en lecture seule de la source) ou bidirectionnelle (TwoWay), ce qui permet dans ce dernier cas de mettre à jour la source.

**Activité 3 : Les différentes étapes de la connexion de l’Access 2010 et VB2010 express**

* Je donne les étapes à suivre pour se connecter à la Base de Données :
* Lancer VB2010 Express ;
* Cliquer sur le nouveau projet dans VB;
* Cliquer sur Application Windows Forms ;
* Taper le nom du projet;
* Cliquer sur ok ;
* Boite à outil ;
* Cliquer sur DataGridView ;
* Vous trouverez DataGridView dans un form ;
* Cliquer en haut à droite de DataGridView ;
* Cliquer sur choisir la source de données ;
* Cliquer sur ajouter de données du projet ;
* Cliquer sur la Base de données ;
* Cliquer sur Suivant ;
* Cliquer sur Dataset ;
* Cliquer sur suivant ;
* Cliquer sur la nouvelle connexion ;
* Cliquer sur modifier ;
* Choisir le fichier de la base de données Microsoft Access ;
* Cliquer sur ok ;
* Cliquer sur parcourir pour chercher le nom de la base de données ;
* Cliquer sur le nom de la base de données ;
* Cliquer sur Tester la connexion ;
* Vous touverez le message « le test de la connexion a réussi » ;
* Cliquer ok…ok ;
* Cliquer sur suivant ;
* Cliquer sur oui ;
* Cliquer sur table et selectionner tous les noms des champs ;
* Cliquer sur terminer ;
* En fin DataGridView affiche dans un form ;

**Activité 4 : décrire les méthodes de la connexion à une base de données Access**

* **Première méthode :**
* Lancer VB2010 express ;
* Cliquez sur l’onglet « Données » ;
* Ajouter une nouvelle source de données… ;
* Base de données ;
* Cliquer sur suivant ;
* Cliquer sur la nouvelle connexion ;
* Cliquer sur modifier ;
* Choisir le fichier de la base de données Microsoft Access ;
* Cliquer sur ok ;
* Cliquer sur parcourir pour chercher le nom de la base de données ;
* Cliquer sur le nom de la base de données ;
* Cliquer sur Tester la connexion ;
* Vous touverez le message « le test de la connexion a réussi » ;
* Cliquer ok…ok ;
* Cliquer sur suivant ;
* Cliquer sur oui ;
* Cliquer sur table et selectionner tous les noms des champs ;
* Cliquer sur terminer ;
* En fin DataGridView affiche dans un form ;**……(cfr.** les étapes à suivre pour se connecter à la Base de Données)
* **Deuxième méthode :**
* **Lancer VB2010 express ;**
* **Boîte à outils ;**
* **DataGridView ;**
* **Choisir la source de données ;**
* **Ajouter la source de données du projet;**
* **Base de données ;**
* Cliquer sur suivant ;
* Cliquer sur la nouvelle connexion ;
* Cliquer sur modifier ;
* Choisir le fichier de la base de données Microsoft Access ;
* Cliquer sur ok ;
* Cliquer sur parcourir pour chercher le nom de la base de données ;
* Cliquer sur le nom de la base de données ;
* Cliquer sur Tester la connexion ;
* Vous touverez le message « le test de la connexion a réussi » ;
* Cliquer ok…ok ;
* Cliquer sur suivant ;
* Cliquer sur oui ;
* Cliquer sur table et selectionner tous les noms des champs ;
* Cliquer sur terminer ;
* En fin DataGridView affiche dans un form ;

1. **Je retiens l’essentiel.**

|  |
| --- |
| * Les liaisons de données sont des processus qui établissent une connexion entre l'interface utilisateur de l'application et le logique métier. En d'autres termes, elles permettent d'établir une connexion entre un contrôle et une source de données.   La création d’un nouveau projet   * Lancer VB2010 Express ; * Cliquer sur le nouveau projet dans VB; * Cliquer sur Application Windows Forms ; * Taper le nom du projet; |

**……………………………………………………………………………………………………………………………………..**

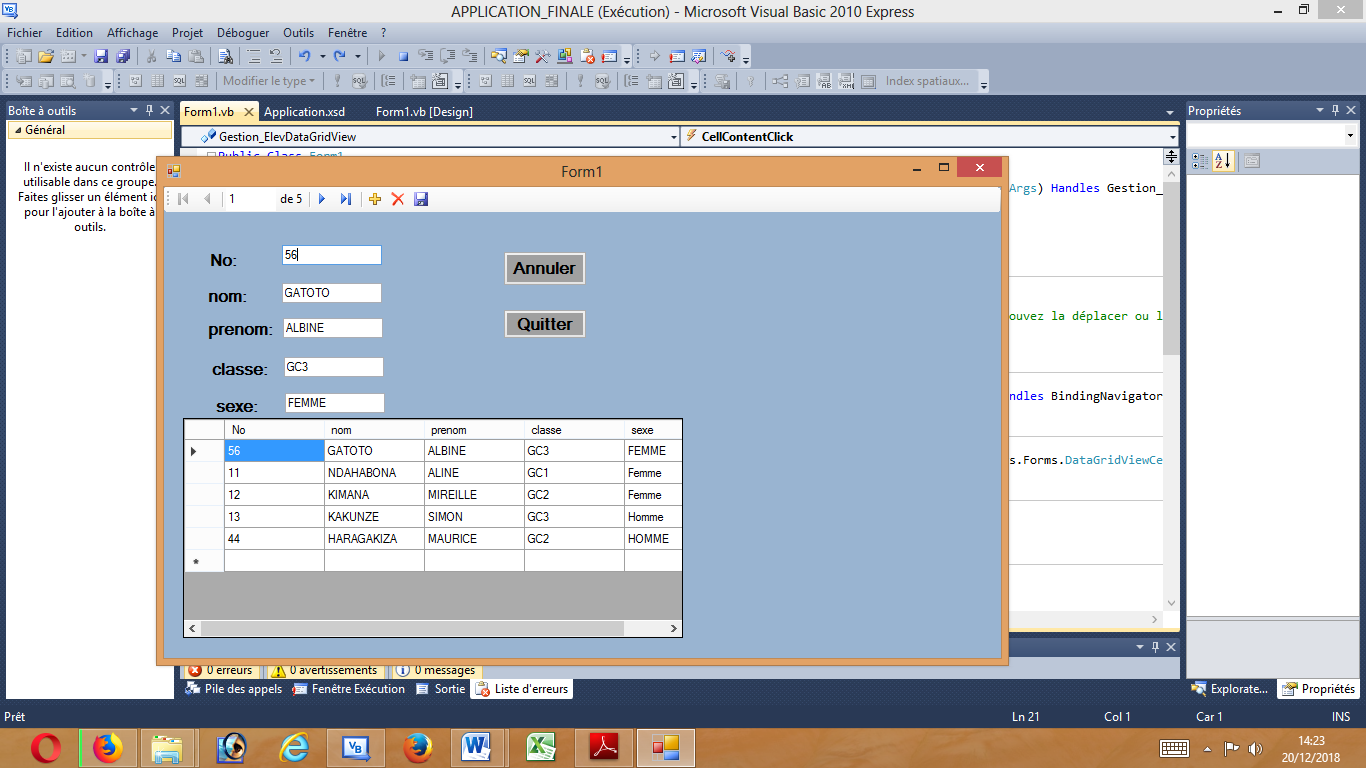
1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**

**Après avoir définir binding, donner son rôle.**

**Binding** est un processus qui établit une connexion entre l'interface utilisateur de l'application et le logique métier.

Son rôle est d'établir une connexion entre un contrôle et une source de données.

* 1. **En utilisant mes connaissances**



Créer une base donne nommée «Gestion.Elèves», enregistrée sur un disque locale D contient une table nommée «Gestion\_Elève».

Créer un projet avec VB 2010 express sous le de «Application\_Finale» et mettez un formulaire, confère image si dessus.

**Reponses:**

Public Class Form1

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

Me.Validate()

Me.Gestion\_ElevBindingSource.EndEdit()

Me.TableAdapterManager.UpdateAll(Me.\_Gestion\_ElevesDataSet)

End Sub

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

'TODO: cette ligne de code charge les données dans la table '\_Gestion\_ElevesDataSet.Gestion\_Elev'. Vous pouvez la déplacer ou la supprimer selon vos besoins.

Me.Gestion\_ElevTableAdapter.Fill(Me.\_Gestion\_ElevesDataSet.Gestion\_Elev)

End Sub

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

End Sub

Private Sub Gestion\_ElevDataGridView\_CellContentClick(ByVal sender As System.Object, ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles Gestion\_ElevDataGridView.CellContentClick

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

End

End Sub

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

Button1.Text = "Annuler"

NoTextBox.Text = ""

NomTextBox.Text = ""

PrenomTextBox.Text = ""

ClasseTextBox.Text = ""

SexeTextBox.Text = ""

End Sub

Private Sub NoTextBox\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles NoTextBox.TextChanged

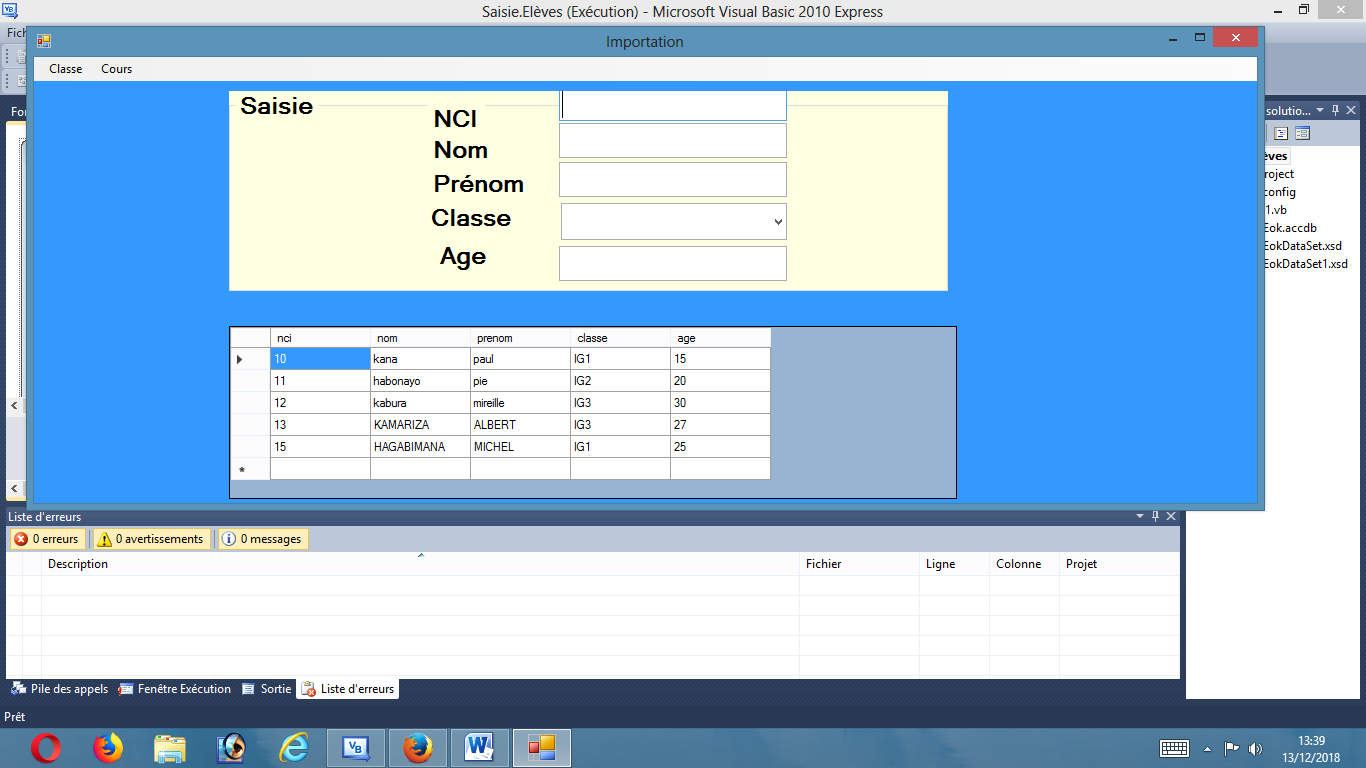
End Sub

End Class

**Pour aller plus loin**

*Ecrire un programme permettant de demander le nombre des apprenant de votre classe, d’entrer leurs notes de l’interrogation de mathématique sur 20, afficher la sommes, la moyenne, le maximum et le minimum des notes.*

D’où alors vous trouverez des données de l’access2010 dans l’interface de VB2010 Express:



**Leçon 23 : Gestion des données sous VB**

* Lancer VB 2010 ;
* Cliquer sur « Données » ;
* Cliquer sur « Base de données» ;
* Cliquer sur « Suivant» ;
* Cliquer sur « DataSet» ;
* Cliquer sur «Suivant» ;
* Cliquer sur « Nouvelle connexion» ;
* Cliquer sur «Modifier» ;
* Cliquer sur «Fichier BD MS Access» ;
* Cliquer sur «Parcourir» ;
* Cliquer sur «Chercher le nom de la base de données» ;
* Cliquer sur «Tester la connexion» ;
* Cliquer sur «Ok» ;
* Cliquer sur «Ok» ;
* Cliquer sur «Suivant» ;
* Cliquer sur «Oui» ;
* Cliquer sur «Suivant» ;
* Cliquer sur «Table, Vue» ;
* Cliquer sur «Terminer».

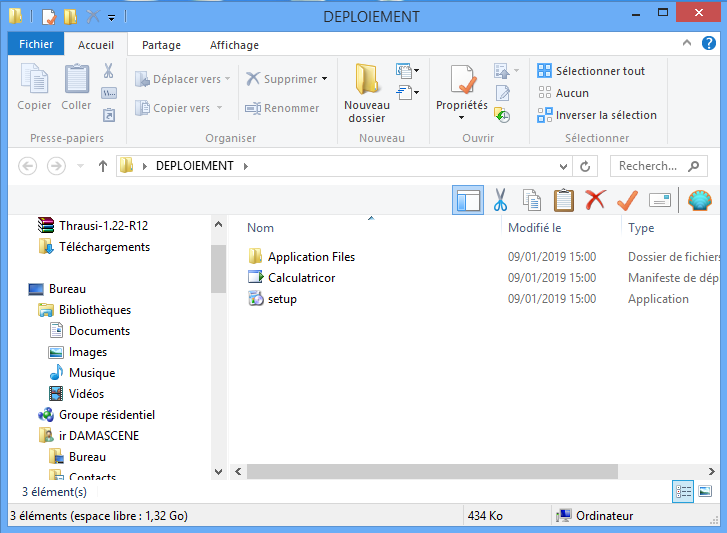
**Objectifs pédagogiques 3 : Créer un fichier exécutable**

**Leçon 24 : Empaquetage et déploiement**

1. **Je saurai:**

* Empaqueter une application;
* Déployer une application ;
* Installer une application déployée ;
* Lancer une application déjà installée.

1. **J’observe et je réfléchi:**



**Image 24.1 Montrer un dossier contenant un fichier exécutable nommé setup de type application.**

**Activité 1 : Observation**

Je découvre que sous MS VB 2010 express nous pouvons créer des applications installable sous Windows.

**Activité 2 : Empaquetage d’une application**

L’empaquetage d’une application consiste à créer un répertoire qui stockera les fichiers qu’on va déployer sous MS VB.

Exemple : Créer un dossier au bureau sous le de : «**DEPLOIEMENT**» qui recevra les fichiers qu’on va déployer.

**Activité 3 : Déploiement d’une application sous MS VB2010 express**

**Je donne les étapes de déploiement d’un fichier**

* Je donne les étapes à suivre pour se connecter à la Base de Données :
* Lancer VB2010 Express ;
* Cliquer sur le nouveau projet dans VB;
* Cliquer sur Application Windows Forms ;
* Taper le nom du projet;
* Cliquer sur ok ;
* Boite à outil ;
* Cliquer sur DataGridView ;
* Vous trouverez DataGridView dans un form ;
* Cliquer en haut à droite de DataGridView ;
* Cliquer sur choisir la source de données ;
* Cliquer sur ajouter de données du projet ;
* Cliquer sur la Base de données ;
* Cliquer sur Suivant ;
* Cliquer sur Dataset ;
* Cliquer sur suivant ;
* Cliquer sur la nouvelle connexion ;
* Cliquer sur modifier ;
* Choisir le fichier de la base de données Microsoft Access ;
* Cliquer sur ok ;
* Cliquer sur parcourir pour chercher le nom de la base de données ;
* Cliquer sur le nom de la base de données ;
* Cliquer sur Tester la connexion ;
* Vous touverez le message « le test de la connexion a réussi » ;
* Cliquer ok…ok ;
* Cliquer sur suivant ;
* Cliquer sur oui ;
* Cliquer sur table et selectionner tous les noms des champs ;
* Cliquer sur terminer ;
* En fin DataGridView affiche dans un form ;

**Activité 4 : décrire les méthodes de la connexion à une base de données Access**

* **Première méthode :**
* Lancer VB2010 express ;
* Cliquez sur l’onglet « Données » ;
* Ajouter une nouvelle source de données… ;
* Base de données ;
* Cliquer sur suivant ;
* Cliquer sur la nouvelle connexion ;
* Cliquer sur modifier ;
* Choisir le fichier de la base de données Microsoft Access ;
* Cliquer sur ok ;
* Cliquer sur parcourir pour chercher le nom de la base de données ;
* Cliquer sur le nom de la base de données ;
* Cliquer sur Tester la connexion ;
* Vous touverez le message « le test de la connexion a réussi » ;
* Cliquer ok…ok ;
* Cliquer sur suivant ;
* Cliquer sur oui ;
* Cliquer sur table et selectionner tous les noms des champs ;
* Cliquer sur terminer ;
* En fin DataGridView affiche dans un form ;**……(cfr.** les étapes à suivre pour se connecter à la Base de Données)
* **Deuxième méthode :**
* **Lancer VB2010 express ;**
* **Boîte à outils ;**
* **DataGridView ;**
* **Choisir la source de données ;**
* **Ajouter la source de données du projet;**
* **Base de données ;**
* Cliquer sur suivant ;
* Cliquer sur la nouvelle connexion ;
* Cliquer sur modifier ;
* Choisir le fichier de la base de données Microsoft Access ;
* Cliquer sur ok ;
* Cliquer sur parcourir pour chercher le nom de la base de données ;
* Cliquer sur le nom de la base de données ;
* Cliquer sur Tester la connexion ;
* Vous touverez le message « le test de la connexion a réussi » ;
* Cliquer ok…ok ;
* Cliquer sur suivant ;
* Cliquer sur oui ;
* Cliquer sur table et selectionner tous les noms des champs ;
* Cliquer sur terminer ;
* En fin DataGridView affiche dans un form ;

1. **Je retiens l’essentiel.**

|  |
| --- |
| * Les liaisons de données sont des processus qui établissent une connexion entre l'interface utilisateur de l'application et le logique métier. En d'autres termes, elles permettent d'établir une connexion entre un contrôle et une source de données.   La création d’un nouveau projet   * Lancer VB2010 Express ; * Cliquer sur le nouveau projet dans VB; * Cliquer sur Application Windows Forms ; * Taper le nom du projet; |

**……………………………………………………………………………………………………………………………………..**

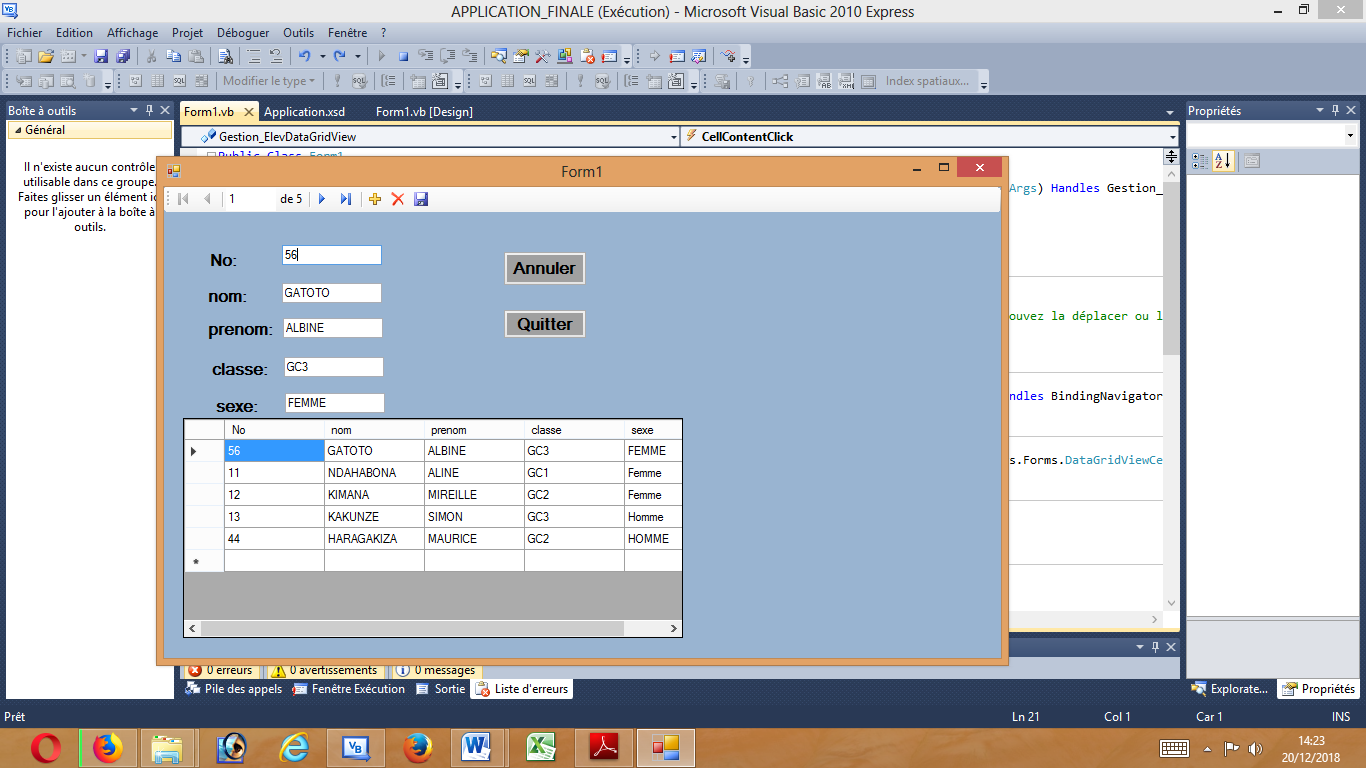
1. **Je m’entraine et je m’évalue**
   1. **En restituant la leçon**

**Après avoir définir binding, donner son rôle.**

**Binding** est un processus qui établit une connexion entre l'interface utilisateur de l'application et le logique métier.

Son rôle est d'établir une connexion entre un contrôle et une source de données.

* 1. **En utilisant mes connaissances**



Créer une base donne nommée «Gestion.Elèves», enregistrée sur un disque locale D contient une table nommée «Gestion\_Elève».

Créer un projet avec VB 2010 express sous le de «Application\_Finale» et mettez un formulaire, confère image si dessus.

**Reponses:**

Public Class Form1

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

Me.Validate()

Me.Gestion\_ElevBindingSource.EndEdit()

Me.TableAdapterManager.UpdateAll(Me.\_Gestion\_ElevesDataSet)

End Sub

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

'TODO: cette ligne de code charge les données dans la table '\_Gestion\_ElevesDataSet.Gestion\_Elev'. Vous pouvez la déplacer ou la supprimer selon vos besoins.

Me.Gestion\_ElevTableAdapter.Fill(Me.\_Gestion\_ElevesDataSet.Gestion\_Elev)

End Sub

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

End Sub

Private Sub Gestion\_ElevDataGridView\_CellContentClick(ByVal sender As System.Object, ByVal e As System.Windows.Forms.DataGridViewCellEventArgs) Handles Gestion\_ElevDataGridView.CellContentClick

End Sub

Private Sub Button2\_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click

End

End Sub

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

Button1.Text = "Annuler"

NoTextBox.Text = ""

NomTextBox.Text = ""

PrenomTextBox.Text = ""

ClasseTextBox.Text = ""

SexeTextBox.Text = ""

End Sub

Private Sub NoTextBox\_TextChanged(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles NoTextBox.TextChanged

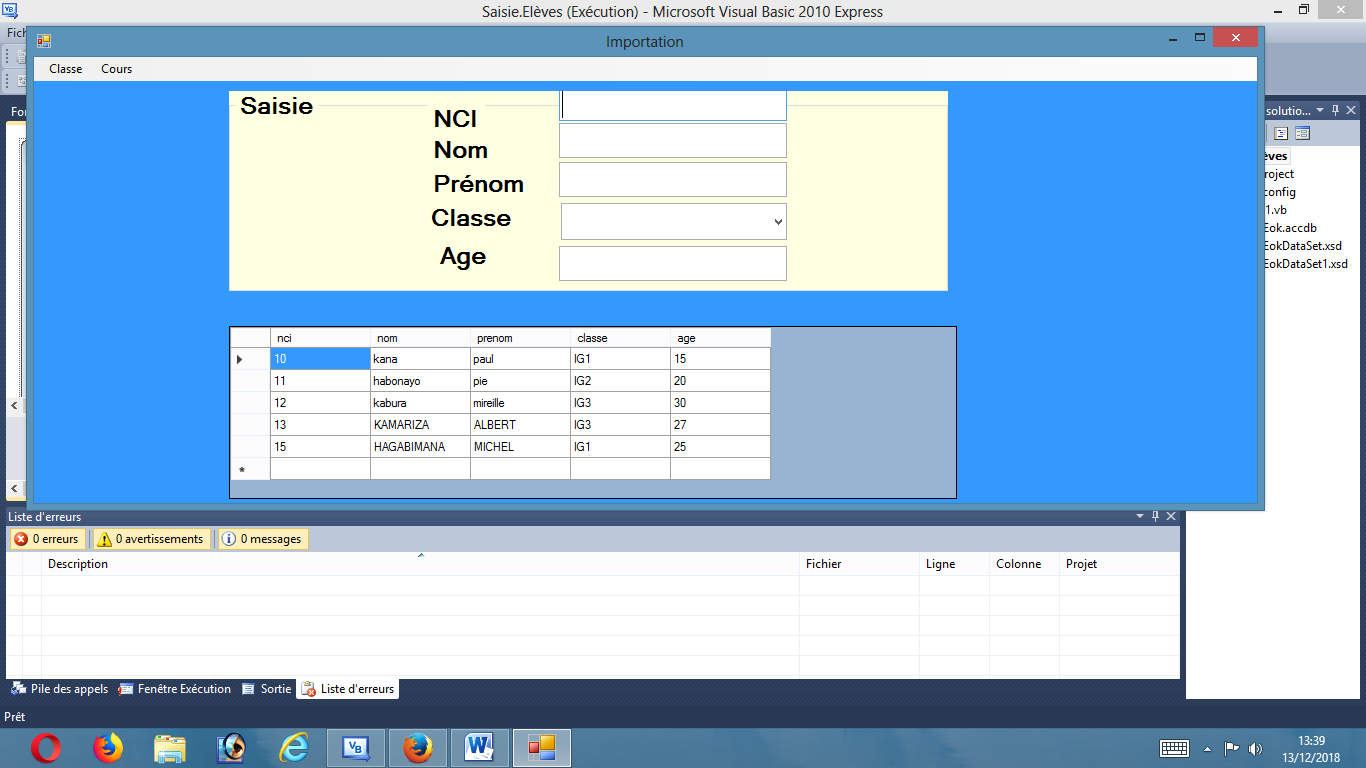
End Sub

End Class

**Pour aller plus loin**

*Ecrire un programme permettant de demander le nombre des apprenant de votre classe, d’entrer leurs notes de l’interrogation de mathématique sur 20, afficher la sommes, la moyenne, le maximum et le minimum des notes.*

D’où alors vous trouverez des données de l’access2010 dans l’interface de VB2010 Express:



**Leçon 26 : Exercices d’application (Créer une application de la gestion d’un stock)**

[Compétence : Développer un programme informatique 1](#_Toc534124807)

[**Elément de la compétence 1 : Maitriser les connaissances en Visual Basic** 1](#_Toc534124808)

[**Objectifs pédagogiques 1 : Découvrir l’environnement Visual Basic** 1](#_Toc534124809)

[**Leçon n°1 : Introduction sur le Visual Basic.** 1](#_Toc534124810)

[**Leçon n°2 : Description de l’interface VB**. 5](#_Toc534124811)

[**Objectif pédagogique 2 : Décrire les éléments de l’application VB** 12](#_Toc534124812)

[**Leçon n°3 : Etude d’une boite à outils**. 12](#_Toc534124813)

[**Leçon n°4: Découvrir les fenêtres de propriétés**. 17](#_Toc534124814)

[**Leçon n°5 : Distinction et déclaration de variables en VB**. 20](#_Toc534124815)

[**Leçon n°6 : Distinction de types de modules en VB**. 26](#_Toc534124816)

[**Élément de la compétence : Ecrire un programme en Visual Basic.** 30](#_Toc534124817)

[**Objectif pédagogique 1 : Utiliser les opérateurs.** 30](#_Toc534124818)

[**Leçon 7 : Distinction des types des opérateurs.** 30](#_Toc534124819)

[**Leçon 8 : Utilisation des variables, des opérations et constantes.** 34](#_Toc534124820)

[**Objectif pédagogique 2 : connaitre les instructions VB** 39](#_Toc534124821)

[**Leçon 9 : Distinction des structures des contrôles.** 39](#_Toc534124822)

[**Leçon 10 : Utilisation du structure de contrôle conditionnelle Si (If).** 45](#_Toc534124823)

[**Leçon 11 : Utilisation de la structure de contrôle Select Case** 54](#_Toc534124824)

[**Leçon 12 : Utilisation de la structure de contrôle du Do Loop.** 58](#_Toc534124825)

[**Leçon 13 : Utilisation de la structure de contrôle While … end while.** 63](#_Toc534124826)

[**Leçon 14 : Utilisation de la structure de contrôle For … Each.** 65](#_Toc534124827)

[**Leçon 15 : Utilisation de la structure de contrôle For … Next.** 69](#_Toc534124828)

[**Leçon 16 : Exercices d’auto-évaluations.** 73](#_Toc534124829)

[**Objectif pédagogique 3.Utiliser les procédures et les paramètres** 81](#_Toc534124830)

[**Leçon 17 : Distinction des procédures** 81](#_Toc534124831)

[**Leçon 18 : Distinction des paramètres** 88](#_Toc534124832)

[**Leçon 19 : Différents types de classe** 94](#_Toc534124833)

[**Leçon 20 : Ajout et suppression d’une Référence** 100](#_Toc534124834)

[**Objectif pédagogique 4 : Créer l’interface utilisateur** 103](#_Toc534124835)

[**Leçon 21 : Création d’une fenêtre VB** 103](#_Toc534124836)

[**Elément de la compétence 3 : Gérer les données** 104](#_Toc534124837)

[**Objectifs pédagogiques 1 : Accéder aux données** 104](#_Toc534124838)

[**Leçon 22 : Rappel sur la création et la gestion d’une base de données** 104](#_Toc534124839)

[**Objectifs pédagogiques 2 : Manipuler les fichiers** 108](#_Toc534124840)

[**Leçon 23 : Connection d’une application VB à une BD** 108](#_Toc534124841)

[**Leçon 24 : Gestion des données sous VB** 113](#_Toc534124842)

[**Objectifs pédagogiques 3 : Créer un fichier exécutable** 114](#_Toc534124843)

[**Leçon 25 : Empaquetage et déploiement** 114](#_Toc534124844)

[**Leçon 26 : Exercices d’application (Créer une application de la gestion d’un stock)** 115](#_Toc534124845)