Le premier programme que vous avez créé dans le chapitre précédent est complet, mais il ne réalise rien de réellement important ou d'utile. Vous allez donc à présent donner à ce programme une réelle dynamique, grâce à l'utilisation de variables.

Qu'est-ce qu'une variable ? Il s'agit tout simplement d'un petit espace que vous réservez à l'intérieur de la mémoire de votre ordinateur pour y stocker une valeur d'un type donné. Bien évidemment, il n'est pas utile de connaître le fonctionnement interne de votre machine pour utiliser des variables.

En fait, la mémoire de la machine fonctionne exactement comme la vôtre. Par exemple, pour mémoriser le numéro de téléphone de Sylvie, votre cerveau alloue un petit espace dans lequel il stocke 06.12.34.56.78, et l'associe à Sylvie en se rappelant qu'il s'agit d'un numéro de téléphone.

Pour stocker une variable dans la mémoire d'un ordinateur, il faut raisonner de la même manière, c'est-à-dire qu'il faut lui donner un nom unique et indiquer son type (nombre, chaîne de caractères…).

Déclarer une variable

Lancez Visual Basic Express et créez un nouveau projet. Appelez-le `ProjetVariables`. Une fois dans le designer, double-cliquez sur la fenêtre pour lancer l'éditeur de code. Une fois dans le code, ajoutez ces deux lignes à l'endroit où est placé le curseur :

```
Dim num As Integer
  num = 100
```

Avec ses deux lignes, vous précisez à l'ordinateur qu'il doit stocker une variable de type nombre entier. Vous lui donnez la valeur 100 et vous l'appelez `num`. Examinons le contenu de ce code :

- `Dim` : ce mot-clé du langage permet de spécifier que vous déclarez une variable.
- `num` : c'est le nom de la variable.
- `As` : ce mot-clé permet de spécifier le type de la variable utilisée.
- `Integer` : il s'agit du type nombre entier.
- `num = 100` : cette opération affecte à la variable `num` la valeur 100.

Vous avez à présent dans le programme une variable nommée `num` qui contient la valeur 100. C'est aussi simple que cela. Voyons maintenant comment l'utiliser.

2.2. Utiliser une variable

Pour utiliser une variable dans un programme, il suffit de placer son nom à l'endroit où vous désirez utiliser sa valeur. Pour l'exemple du numéro de téléphone, il suffirait d'écrire "Sylvie" et l'ordinateur saurait qu'il faut remplacer ce libellé par 06.12.34.56.78. Ainsi, l'exemple suivant affiche la valeur de la variable `num`. Replacez-vous dans l'éditeur de code et ajoutez ces deux lignes :

```
num = num+100
MessageBox.Show(num.ToString)
```

Appuyez maintenant sur F5. Le programme se lance, avec une boîte de dialogue qui donne la valeur de `num`, à savoir 200.

Ces deux lignes permettent d'afficher la nouvelle valeur de `num`, à laquelle vous avez ajouté 100. L'opération `ToString` permet de transformer un type nombre en chaîne de caractères en vue de son affichage. Il existe un grand nombre d'opérations possible utilisant les variables. Il est maintenant temps de se familiariser avec les différents types de variables.

Parmi ceux-ci existent les booléens, qui permettent de raisonner sous forme de conditions, les nombres entiers et décimaux, qui permettent d'effectuer des calculs, et les caractères et chaînes de caractères, qui permettent de manipuler du texte. Chacun de ces types est incompatible avec les autres et vous ne pourrez donc pas additionner un nombre avec du texte par exemple. Comme vous l'avez vu plus haut, il est possible de convertir un type en un autre à l'aide de méthodes déjà existantes. C'est le cas de `num.ToString`, qui transforme le nombre `num` en chaîne de caractères. Voici le détail de chaque type de variable et de ce qu'il permet.

Les booléens

Les booléens sont des variables issues de la logique du mathématicien George Boole. Cette logique permet une représentation par état, qui donne le passage du courant comme étant égal à 1 et le non-passage du

courant comme étant égal à 0. Elle est à la base de l'électronique et de l'informatique.

Dans une réflexion de programmation, déclarer une variable booléenne permet de réagir selon certains états. Dans chaque variable booléenne déclarée, on peut stocker un état, par exemple si une porte est ouverte ou non. Un booléen ne peut prendre que deux valeurs : "vrai" ou "faux".

Voici pour la mise en pratique :

1. Ouvrez un nouveau projet avec Visual Basic Express et nommez-le `ProjetBool`. Puis, dans le designer, placez-vous dans l'éditeur de code en double-cliquant sur la fenêtre.
2. À l'endroit du curseur, saisissez le code `Dim monBoo As Boolean`, ce qui a pour effet de déclarer une variable de type booléenne, de nom `monBoo`.
3. Retournez à la ligne et saisissez `monBoo =`. Lorsque vous entrez le caractère =, l'éditeur de texte vous propose d'attribuer une valeur au booléen. Pour cet exemple, choisissez `True`.
4. Ajoutez la ligne `MessageBox.Show(monBoo.ToString).`
5. Lancez le programme en appuyant sur F5.

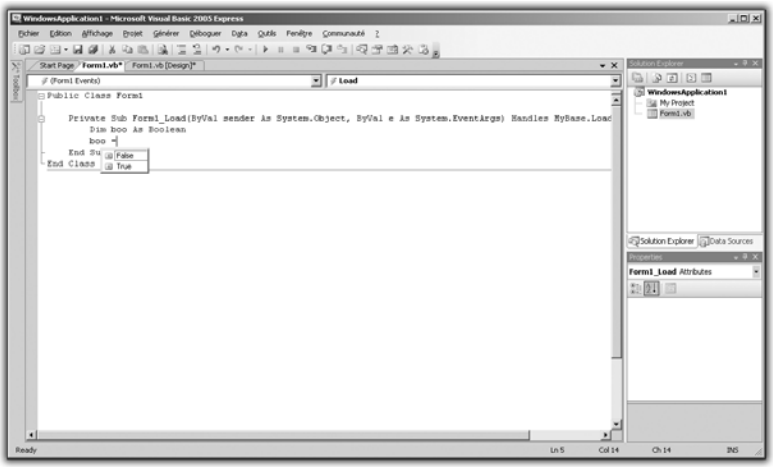

Figure 2.1 : *Affichage du booléen*

L'utilisation de telles variables permet de réaliser des tests dans les applications. Pour votre première application, il aurait par exemple été

pertinent d'utiliser un booléen pour vérifier que l'utilisateur a bien entré son prénom avant d'appuyer sur le bouton OK.

Vous verrez comment faire cela aux chapitres ***Contrôler un programme*** *et* ***Dialoguer avec un ordinateur****.*

2.4. Les nombres

Les nombres entiers

Que vous réalisiez une application de manipulation d'images, de comptabilité, de calendrier ou d'annuaires, vous aurez forcément besoin de manipuler des nombres. Nous avons déjà vu en introduction de ce chapitre que vous pouviez initialiser une variable en tant que type numérique entier déclaré à l'aide du mot-clé `Integer`.

En plus de la possibilité de déclarer des variables de type entier, vous pouvez réaliser toute une série d'opérations sur ces nombres : par exemple l'addition, la soustraction, la multiplication, la division, la division entière, etc.

Pour initialiser une variable de type nombre entier, utilisez la commande suivante :

```
Dim MaVariable As Integer
MaVariable = 5
```

Cela fait, la variable nommée `MaVariable` possède la valeur 5. Initialisez maintenant une nouvelle variable de type entier :

```
Dim MaVariable2 As Integer
MaVariable2 = 3
```

Pour tester les opérations possibles entre ces deux variables, vous allez initialiser une troisième variable de type entier, qui permettra de stocker le résultat d'une opération :

```
Dim MonResultat As Integer
MonResultat = MaVariable + MaVariable2
```

Ici les deux valeurs sont additionnées. Libre à vous de changer le caractère `+` pour `*`, qui réalise la multiplication des deux valeurs, ou encore `/`, qui réalise une division.

Affichez maintenant le résultat :

```
MessageBox.Show(MonResultat.ToString)
```

Pour tester cet exemple, créez un nouveau projet dans Visual Studio Express, puis double-cliquez sur la fenêtre dans le designer. Une fois dans l'éditeur de code, saisissez les instructions précédentes. Lancez le programme avec F5.

Figure 2.2 :
Résultat de l'addition de deux nombres

Ajouter des virgules

Nous avons vu le cas des nombres entiers. Mais que se passe-t-il si vous stockez une variable contenant le résultat de la division de deux entiers dans un autre nombre entier ? Si la division tombe juste, tout va bien. Par contre, si le résultat est un nombre à virgule, plus rien ne marche !

En effet, l'ordinateur n'étant doué d'aucune intelligence, il ne comprend pas comment faire entrer un nombre d'un certain type dans une variable qui n'est pas de ce type. Il faut donc recourir à un autre type de variable : `Decimal`.

Voici un exemple de programme que vous pouvez reprendre par exemple pour calculer une moyenne :

```
Dim x As Integer
x = 10
Dim y As Integer
y = 14
Dim z As Integer
z = 8
Dim res As Decimal
res = ((10+14+8)/3)
MessageBox.Show(res.ToString)
```

Figure 2.3 :
Résultat de la moyenne

Déclarer une variable tout en l'initialisant

Lorsque vous déclarez des variables, il n'est pas utile de préciser le type si vous faites tout de suite l'initialisation. Lorsque le programme sera compilé, le compilateur va automatiquement reconnaître le type de variable. Par exemple, `Dim x = 42` est équivalent à `Dim x As Integer` suivi de `x = 42`.

2.5. Les jeux de lettres

Les caractères

Après les chiffres, voyons les variables de type caractère et chaîne de caractères.

Un caractère peut être un chiffre, une lettre, l'apostrophe, la parenthèse ou encore le retour chariot et même l'espacement.

Pour déclarer une variable de type caractère, utilisez la syntaxe suivante :

```
Dim c As Char
c = 'f'
```

Notez que les caractères sont initialisés à l'aide d'une valeur entre apostrophes. C'est le cas du `'f'` dans cet exemple.

Les variables de type caractère peuvent être utiles dans le cadre de travaux sur les mots, par exemple si vous souhaitez vérifier que la première lettre d'un mot est une majuscule. Cela est possible à l'aide d'outils proposés dans Visual Basic Express.

Tout cela sera détaillé au chapitre ***Contrôler un programme.***

Les chaînes

Plusieurs caractères mis les uns après les autres forment une chaîne de caractères. Ces chaînes sont très utilisées en programmation dans la mesure où il faut souvent demander à l'utilisateur d'un programme de

saisir des données. S'il faut traiter ces données sous forme de texte, les chaînes de caractères entrent en jeu.

Pour déclarer une chaîne de caractères, procédez de la manière suivante :

```
Dim S As String
S = "Bonjour"
```

Subtilités

Notez que les chaînes de caractères sont initialisées entre guillemets (`"`), et les caractères entre apostrophes (`'`).

Dans la mesure où l'on travaille souvent sur des chaînes de caractères en programmation, un bon nombre de traitements de base ont déjà été mis au point. Ainsi, pour concaténer deux chaînes de caractères, il suffit de faire comme si vous les additionniez :

```
Dim S1 As String
S1 = "Bonjour"
Dim S2 As String
S2 = " tout le monde"
MessageBox.Show(S1+S2)
```

Figure 2.4 :
Concaténation de deux chaînes

De nombreuses autres méthodes sont disponibles pour le travail sur des chaînes. Par exemple, vous pouvez connaître le nombre de caractères d'une chaîne grâce à l'attribut `length`. Pour l'utiliser, il suffit d'ajouter `.length` à la fin du nom d'une variable de type chaîne :

```
Dim S As String
S = "Bonjour"
MessageBox.Show(S.Length.ToString)
```

Figure 2.5 :
Longueur d'une chaîne

Pour convertir une chaîne de caractères en minuscules ou en majuscules, utilisez les fonctions `ToLower` et `ToUpper`, en ajoutant `.ToUpper` à la fin du nom de votre variable chaîne de caractères. Le code suivant stocke une chaîne convertie en minuscules dans une autre variable de type chaîne de caractères et l'affiche :

```
Dim S As String
S = "BONJOUR"
Dim S2 As String
S2 = S.ToLower
MessageBox.Show(S2)
```

Figure 2.6 : *Chaîne en minuscules*

"BONJOUR" est affiché via `S2` sous forme de "bonjour".

Cas pratique : crypter des messages

Après ces exemples d'utilisation de chaînes, vous allez réaliser une application qui cryptera des messages. Grâce à programme, vous pourrez saisir un message qui ne sera lisible que par quelqu'un disposant de votre application de décryptage.

Dans un premier temps, créez un nouveau projet dans Visual Basic Express. Appelez-le `ProjCryptage` par exemple.

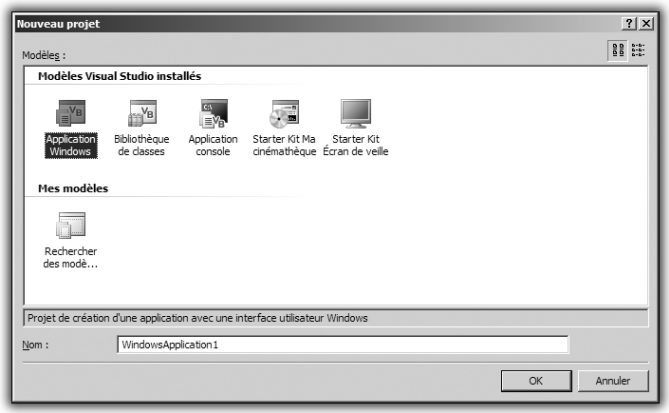

Figure 2.7 : *Création du projet*

Dans le designer, ajoutez à la fenêtre en cours deux *TextBox* et un bouton de sorte que votre fenêtre ressemble à ceci :

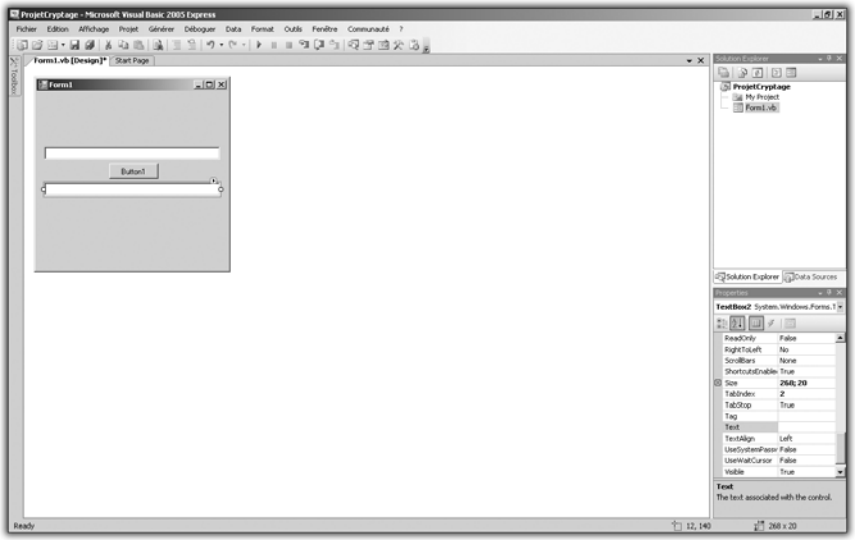

Figure 2.8 : *Design de la fenêtre*

La première *TextBox* est réservée à la saisie du message, la seconde accueillira le message une fois crypté.

Maintenant que le programme est dessiné, il ne reste qu'à écrire le processus de cryptage lorsque l'on clique sur le bouton de l'application.

Pour cela, double-cliquez sur le bouton. Dans l'éditeur de code, à l'endroit où se trouve le curseur, ajoutez les lignes suivantes :

```
Dim s As String
s = TextBox1.Text
s = s.Replace("d", "3")
s = s.Replace("e", "7")
s = s.Replace("g", "1")
s = s.Replace("c", "8")
s = s.Replace("q", "2")
s = s.Replace("h", "g")
s = s.Replace("n", "d")
s = s.Replace("t", "5")
s = s.Replace("j", "6")
s = s.Replace("a", "4")
TextBox2.Text = s
```

Ces lignes d'instruction ont pour effet de copier, dans un premier temps, le contenu de la première *TextBox* dans une variable temporaire. Ensuite, tout le traitement va être fait sur cette variable temporaire. Vous utiliserez la méthode `Replace`, disponible pour toutes les chaînes. Elle permet d'échanger deux caractères dans une chaîne. Une première instruction remplacera toutes les lettres "d" de la phrase par des caractères "3", une seconde tous les "e" par des "7", etc. Toutes ces modifications seront effectuées au moment où vous cliquerez sur le bouton. Une fois le traitement fini, la phrase cryptée apparaîtra dans la deuxième *TextBox*, située sous le bouton.

Ainsi, une phrase telle que "J'ai lancé la cafetière" apparaîtra sous la forme : `"6'4i l4d8é l4 84f75i7r7"`. Bien malin qui pourra retrouver le message d'origine !

Voyons à présent comment décrypter le message.

Rien de plus simple : il suffit de refaire la même application, mais en inversant l'ordre des instructions. Répétez les deux premières étapes de ce projet, puis copiez les instructions suivantes après avoir double-cliqué sur le bouton. Appelez ce projet `ProjDecrypt`.

```
Dim s As String
      s = TextBox1.Text
      s = s.Replace("d", "n")
      s = s.Replace("3", "d")
      s = s.Replace("g", "h")
      s = s.Replace("7", "e")
      s = s.Replace("1", "g")
      s = s.Replace("8", "c")
      s = s.Replace("2", "q")
      s = s.Replace("5", "t")
      s = s.Replace("6", "j")
      s = s.Replace("4", "a")
 TextBox2.Text = s
```

Testez le projet en le lançant avec F5 : la chaîne est bien décryptée !

2.6. Convertir les types

Tout au long de chaque développement, vous devrez transformer des chiffres en chaînes de caractères, et inversement. Même des booléens peuvent être transformés en chaînes ou en types numériques.

Passer d'un entier à une chaîne de caractères

Pour passer d'un entier à une chaîne, il suffit d'utiliser la méthode `ToString`. Elle permet de transformer un entier en un type chaîne utilisable directement.

Voici un exemple d'utilisation de `ToString` :

```
Dim x As Integer
x = 42
MessageBox.Show(x.ToString)
```

Si vous essayez de lancer le programme sans ajouter l'instruction `.ToString` dans `MessageBox.Show(x)`, une erreur surviendra et le programme ne se lancera pas.

Conversion de décimaux
Ce type de conversion marche aussi avec les nombres décimaux.

Transformer une chaîne de caractères en nombre entier

Pour passer d'une chaîne à un nombre, il faut utiliser la méthode `Int32.Parse(votrechaine)`. Elle renvoie un entier qu'il convient de stocker dans une variable appropriée. Voici un exemple d'utilisation :

```
Dim x As Integer
Dim S As String
S = "123456"
x = Int32.Parse(S)
```

Dans ce cas, `x` sera égal au nombre 123 456. Il est nécessaire de réaliser cette conversion si vous désirez effectuer des opérations sur une saisie de l'utilisateur par exemple. Rappelez-vous que l'opérateur + ne veut pas dire la même chose selon que l'on manipule des entiers ou des chaînes de caractères. Examinons le code suivant :

```
Dim s As String
s = "123"
Dim s2 As String
s2 = "456"
MessageBox.Show(s+s2)
```

Il n'affichera pas 579 mais 123 456. Attention donc au type des variables. Pour additionner deux entiers d'abord représentés sous forme de chaînes, vous devez les convertir puis les stocker à part, comme le montre l'exemple suivant :

```
Dim s As String
s = "123"
Dim s2 As String
s2 = "456"
Dim x As Integer
x = Int32.Parse(s)
Dim Y As Integer
Y = Int32.Parse(s2)
Dim Addi As Integer
Addi = x+y
```