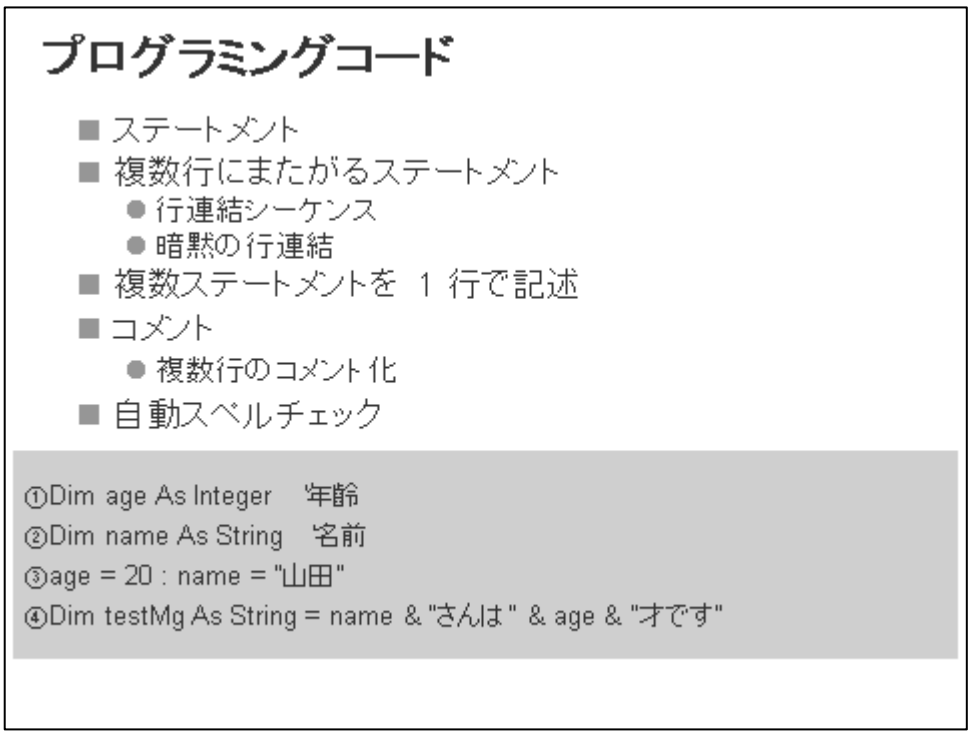


■ ステートメント

Visual Basic では、改行までの1行を1つの意味のあるプログラムコードとして認識し、これをステートメントと呼びます。ステートメントは、左から右、上から下へ実行されます。

■ 複数行にまたがるステートメント

ステートメントが長くなりすぎる場合など、ステートメントを複数行に分割して記述したい場合、行連結ステートメントを使用することができます。行連結ステートメントは半角スペースとアンダースコア(_)を使って表します。Visual Studio 2010 以降では次のような場合は行連結ステートメントを省略可能です。(暗黙の行継続)

- カンマ(,)の後
- 左かっこ(()の後、または右かっこ())の前
- 連結演算子(&)の後
- 代入演算子(=)の後 etc.....

```
1.  MessageBox.Show("メッセージ",
2.  "タイトル")
3.  MessageBox.Show(
4.  GetUserName(ID)
5.  )
6.  cmd.CommandText = "SELECT * FROM employees"&
7.  "WHERE emp_type = 'S'"
8.  Button1.Text =

9.  "OK"
```

■ 複数ステートメントを 1 行で記述

複数ステートメントを1行に記述する場合コロン(:)でステートメントを連結します。

■ コメント

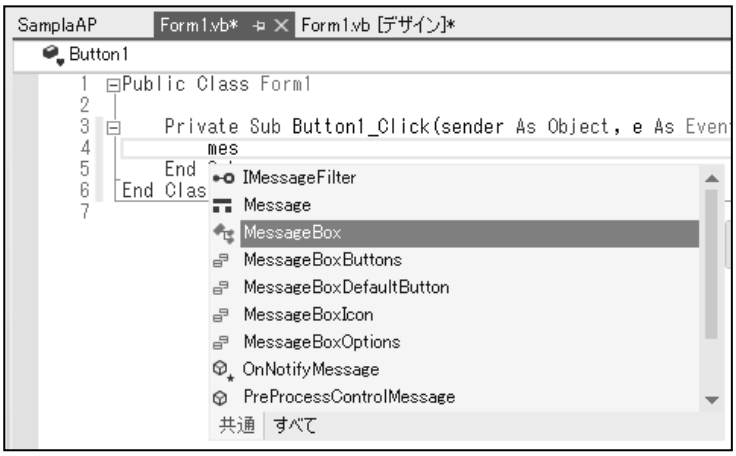
シングルクォーテーション(')からその行の行末までは、コメントとしてあつかわれます。コメントはビルド対象外のコードです。複数行のプログラムコードを一度にコメントアウトする場合、プログラムコードを選択した後、ボタンをクリックします。コメントアウトされたプログラムコードを選択した後、ボタンをクリックすると、コメントを解除することができます。

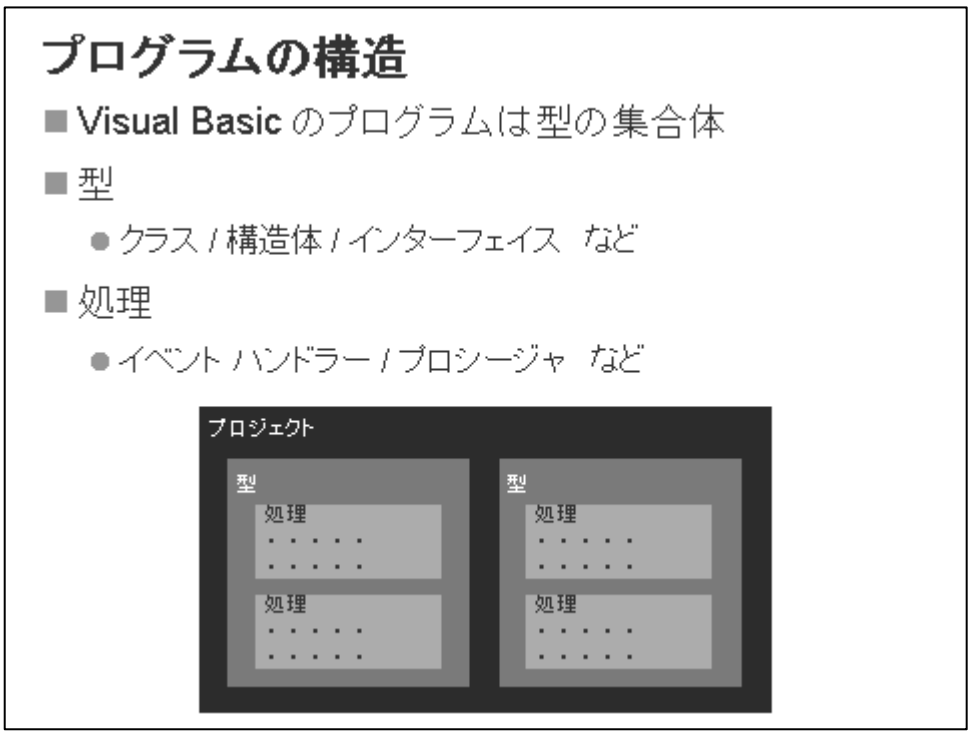
■ 自動スペルチェック

Visual Basic のプログラムコードでは、大文字・小文字を区別しません。しかし、Visual Studio では、入力した文字が、オブジェクト名やキーワードに一致した場合、改行のタイミングで、大文字・小文字の変換処理がおこなわれます。この機能を利用して、プログラムコードをすべて小文字で入力するようにすれば、改行のタイミングで入力ミスを見つけやすくなります。

【インテリセンス】

Visual Studio では、インテリセンス機能によってコード入力時、入力可能なキーワードがプルダウンメニューとして表示されます。マウスのカーソル、または矢印キーで入力したいキーワードを選択し、Tab キーを押すことで入力できます。メニューを半透明にしたい場合は Ctrl キーを、消したい場合は Esc キーを押します。再度表示したい場合は Ctrl キーと J キーを押します。





■ Visual Basic のプログラムは型の集合

Visual Basic で作成されるプログラムはすべて型の集合体(コレクション)として作成されます。処理のみを単独で記述せずに、何らかの型に含めます。

■ 型

型に分類分けされるコードブロックにはクラス、構造体、インターフェイスなどがあります。

■ 処理

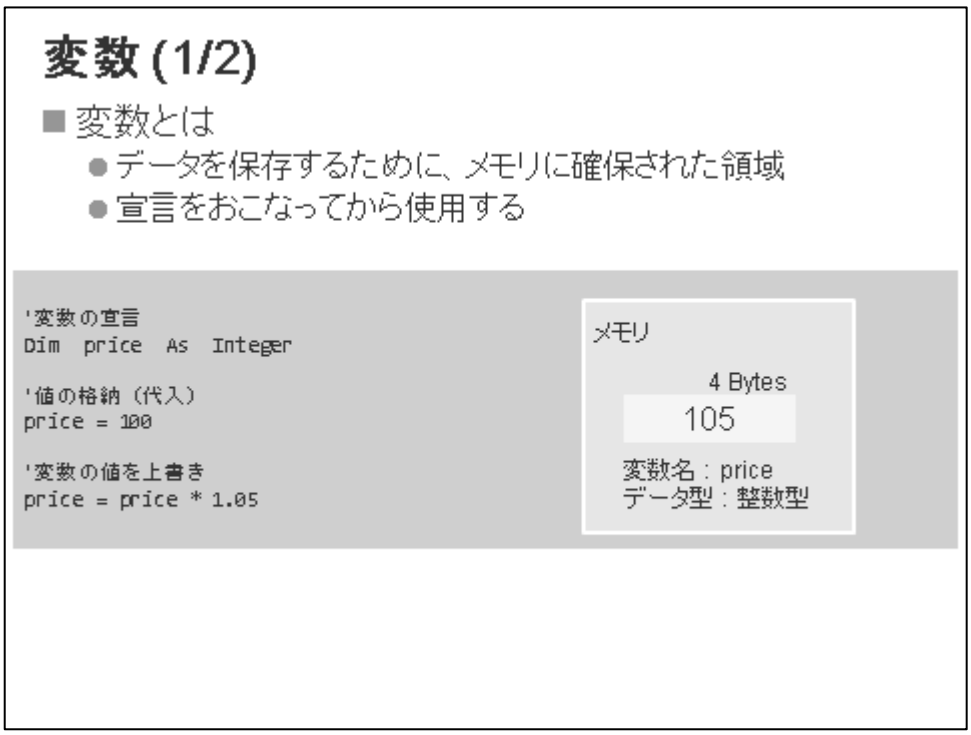
処理に分類分けされるコードブロックにはイベントハンドラー、プロシージャなどがあります。

次のサンプルコードでは、Form1 という名前で型(クラス)を定義し、その中に Button1_Click という名前で処理(イベントハンドラー)を定義しています。

```
1.  Public Class Form1
2.
3.      Private Sub Button1_Click(ByVal sender As System.Object, _
4.          ByVal e As System.EventArgs) Handles Button1.Click
5.
6.          MessageBox.Show("Hello Visual Basic !!")
7.
8.      End Sub
9.
10. End Class
```

■変数とデータ型■

- 変数
- 変数の宣言
- データ型
- 値の代入
- 宣言例
- 配列
- 配列の宣言
- ReDim ステートメント
- データ型の変換
- Option ステートメント
- 定数

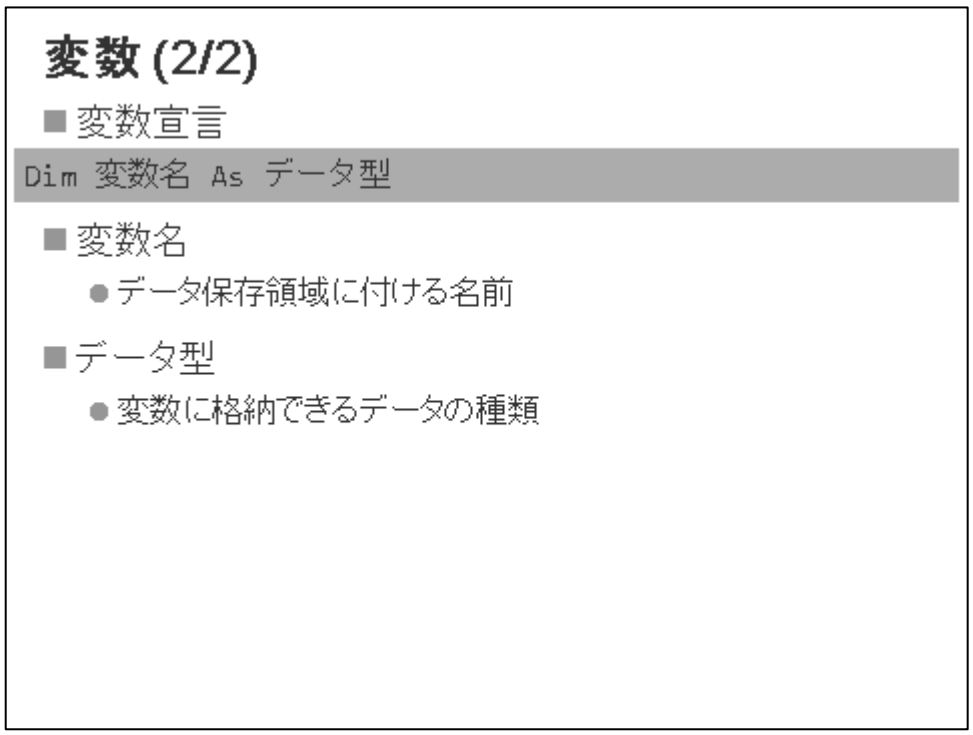


■ 変数とは

変数は、プログラムの中で使用するデータを保存するために、メモリに確保された領域です。変数に格納されたデータは、アプリケーションの実行中に変更することができます。プログラムでは、次のような用途に変数を使用します。

- オブジェクトのプロパティ値の保存
- プロシージャの戻り値の保存
- ユーザーが入力したデータの保存
- 後続処理のため、中途段階での処理結果の一時的な保存
- 同じ処理を複数回繰り返す場合に、現在の回数の保存
- プロシージャのパラメーター　etc.....

```
1.  SelectionName = ListBox1.SelectedItem
2.  AnswerResult = String.Equals(TextBox1.Text, TextBox2.Text)
3.  UserName = InputBox("名前を入力してください")
4.  UserName = TextBox1.Text
5.  NewMessage = "新しいメールが届いています。"
6.  Average = TotalScore / NumberStudent
```



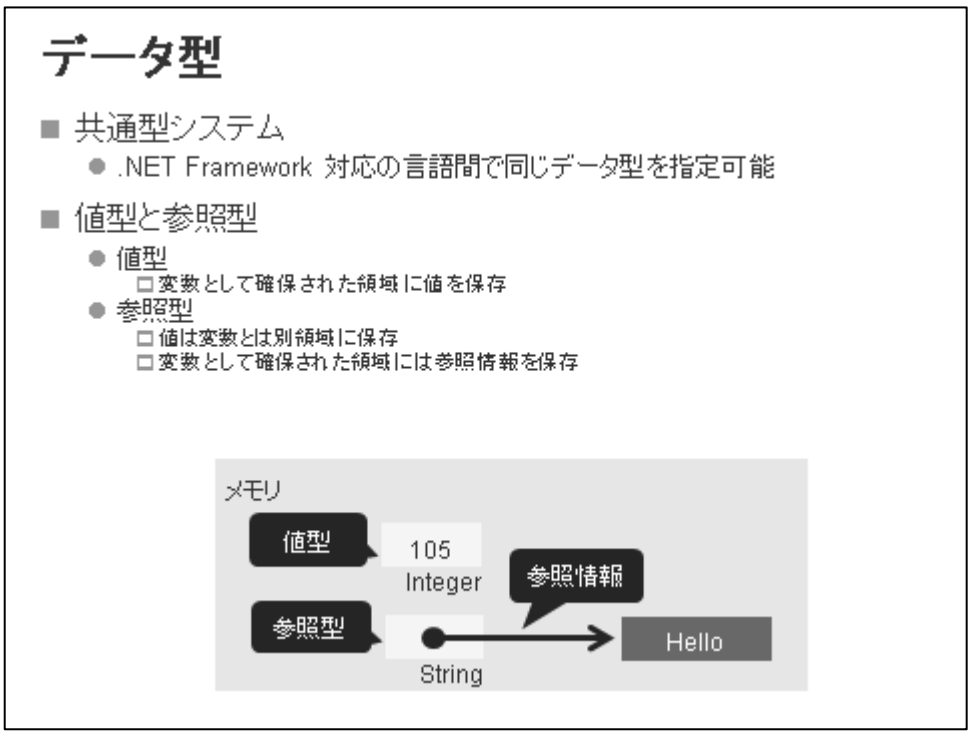
■ 変数名

変数名は、データ保存領域につける名前です。プログラムコードでは、変数名を使って、データ保存領域にデータを格納したり、格納されているデータを参照したりします。変数名は、次の規則内で自由に付けることができます。

- 最初の文字は、英字、アンダースコア(_)のいずれか
- ピリオド(.)、空白、型宣言文字(%、&、!、#、@、$)は、使用できない
- 予約キーワード(ステートメント名、関数名、演算子など)は、使用できない
- 同じスコープ(適用範囲)内で、同じ変数名を複数使用することはできない

■ データ型

As キーワードの後ろにはデータ型を指定します。データ型はその変数にどのような値を格納するのかを表しています。例えば、変数の中に整数の値を格納する場合は Integer というデータ型を指定します。文字列を格納する場合は、String というデータ型を指定します。メモリに確保される領域の大きさは、変数宣言の際に指定するデータ型によって決まります。



■ 共通型システム(CTS：Common Type System)

.NET Framework では、プログラミング言語に依存しない実行環境を実現するため、共通型システムを定義しています。そのため、Visual Basic でも Visual C#でも、同じデータ型を指定して、変数を宣言することができます。また、各プログラミング言語で.NET Framework で定義されたデータ型の別名が定義されているため、各プログラミング言語のデータ型を使用することもできます。

● Visual Basic

```
1.  Dim x AS Integer'Visual Basic のデータ型
2.  Dim x As System.Int32'CTS のデータ型
```

● Visual C#

```
1.  int x;                    //Visual C#のデータ型
2.  System.Int32 x;//CTS のデータ型
```

■ 値型と参照型

データ型は、メモリ内にデータを確保する仕組みの違いから、値型と参照型の 2 つに分類することができます。値型のデータ型で宣言した変数には、その変数に実際のデータが格納されます。参照型のデータ型で宣言した変数には、実際のデータではなく、実際のデータを参照するための参照情報が格納されます。

主なデータ型を次に示します。

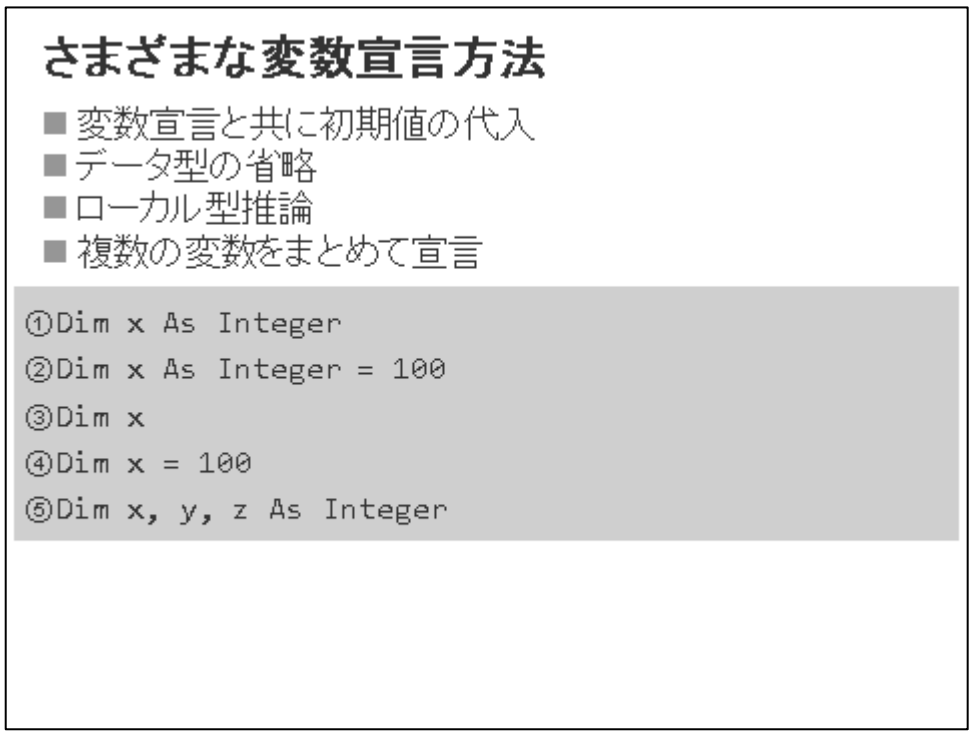
分類 (カテゴリ)	Visual Basic のデータ型	CTS のデータ型	サイズ (byte)	値の範囲
値型 (整数)	Byte	System.Byte	1	0～255
	SByte	System.SByte	1	-128 ～ 127
	Short	System.Int16	2	-32,768 ～ 32,767
	UShort	System.UInt16	2	0～65535
	Integer	System.Int32	4	-2,147,483,648 ～ 2,147,483,647
	UInteger	System.UInt32	4	0 ～ 4,294,967,295
	Long	System.Int64	8	-9,223,372,036,854,775,808～ 9,223,372,036,854,775,807
	ULong	System.UInt64	8	0 ～ 18,446,744,073,709,551,615
値型 (浮動小数点数)	Single	System.Single	4	-3.4028235E+38 ～ -1.401298E-45 (負の値) 1.401298E-45 ～ 3.4028235E+38 (正の値)
	Double	System.Double	8	-1.79769313486231570E+308 ～ -4.94065645841246544E-324 (負の値) 4.94065645841246544E-324 ～ 1.79769313486231570E+308 (正の値)
値型 (その他)	Boolean	System.Boolean	2	True または False
	Char	System.Char	2	Unicode 文字
	Date	System.DateTime	8	西暦 1 年 1 月 1 日～9999 年 12 月 31 日
	Decimal	System.Decimal	16	-79,228,162,514,264,337,593,543,950,335 ～ 79,228,162,514,264,337,593,543,950,335 (小数部分を持たない数値の場合) -7.9228162514264337593543950335 ～ 7.9228162514264337593543950335 (小数点以下 28 桁の数値の場合) 0.0000000000000000000000000001 (1E-28) (0 を除いた場合の絶対値の最小値)
参照型	Object	System.Object	4	任意の型
	String	System.String		Unicode 文字の不変固定長文字列

■ リテラル表記とは

リテラルとはプログラムコード中の定数値のことです。サフィックスを利用することでリテラルのデータ型を指定することができます。サフィックスを省略した場合、整数値は、Integer 型の範囲では Integer 型に、それより大きな値になると Long 型として解釈されます。また実数値は、Double 型で解釈されます。

■ 変数の初期化

変数に値を代入するには、代入演算子(=)を利用します。変数に最初に代入される値のことを初期値と呼び、変数に初期値を代入することを初期化と呼びます。



■ 変数宣言と共に初期値の代入

変数の宣言と初期値の代入をひとつのステートメントでおこなうことができます。

■ データ型の省略

データ型を省略した場合、変数のデータ型は Object 型として取り扱われます。

■ ローカル型の推論

ローカル変数を宣言する場合、データ型を省略し、かつ初期値を代入すると(例：Dim x = 100)代入された値によって適切なデータ型が推論されます。ローカル変数とは、プロシージャ内で宣言された変数のことです。(プロシージャに関しては後述)この機能はローカル型の推論と呼ばれています。ローカル型の推論は Visual Studio 2008 からの機能です。

■ 複数の変数をまとめて宣言

変数宣言をカンマ (,) で区切ることで複数の変数を定義することができます。

■ さまざまな変数宣言例とデータ型

```
1.  Dim x As Integer            'x は Integer 型
2.  Dim x As Integer = 100      'x は Integer 型(初期値：100)
3.  Dim x                       'x は Object 型
4.  Dim x = 100                 'x は型推論されて Integer 型(初期値：100)
5.  Dim x, y, z As Integer      'x,y,z は Integer 型
```