الفصل الرابع

أساسيات البرمجة بلغة Visual Basic

٤.٠ تمهيد

يهدف هذا الفصل إلى إلقاء الضوء على أساسيات البرمجة بشكل عام، والبرمجة بلغة Visual Basic بشكل خاص. ويمكنك ـ عزيزى القارئ ـ اعتبار هذا الفصل البداية الحقيقية لتعلم كيفية كتابة البرامج بلغة VB 2005 Express Edition. يبدأ الفصل بمناقشة أهمية استخدام لغات البرمجة الحديثة فى إمداد الكمبيوتر بالتعليمات. وبعد ذلك نستعرض عددا من الموضوعات الأساسية؛ مثل المتغيرات، وأنواع البيانات، والمصفوفات، وكيفية تنفيذ العمليات الحسابية المختلفة، ثم يختتم الفصل بشرح كيفية عمل المقارنات المنطقية داخل الكود.

الأهداف التعليمية

بنهاية هذا الدرس، يجب أن يكون الدارس قادرا على:

- تعريف المتغيرات Variables وأهميتها.
- تحديد أنواع البيانات Data Types والفرق بين كل منها.
- تمييز المتغيرات النصية String بين غيرها من أنواع المتغيرات.
- إنشاء المصفوفات Arrays
- التحويل بين أنواع البيانات المختلفة
- إجراء العمليات الحسابية Arithmetic Operations المختلفة بلغة VB.NET
- إجراء المقارنات Comparisons بلغة VB.NET

١ـ٤ أسس عامة

ذكرنا من قبل أن الكمبيوتر – فى حد ذاته – ليس آلة ذكية، بل على العكس. فهذا الجهاز بشكل أساسى عبارة عن مجموعة ضخمة من المحولات الكهربية electronic switches التى إما أن تكون فى متصلة On، وإما فى منفصلة Off. ومن خلال الخلط بين هذه المحولات، يمكنك أن تجعل الكمبيوتر يقوم بأداء شيء ما. على سبيل المثال: عرض نص معين على الشاشة، أو إصدار صوت ما. وهذا هو ـ باختصار شديد ـ تعريف البرمجة: أن تقول للكمبيوتر ماذا يفعل.

وبالطبع تعتبر محاولة التعرف على كيفية الخلط بين مجموعة معينة من بين هذا الكم الهائل من المحولات الكهربية مهمة صعبة للغاية بالنسبة إلى الإنسان. ومن ثم تأتى أهمية لغة البرمجة.

فالبشر يستخدمون لغة للتعبير عن أنفسهم بواسطة الكلمات. فى حين أن الكمبيوتر لا يتحدث إلا لغة الأصفار 0s (أو حالة التوقف Off)، والآحاد 1s (أو حالة التشغيل On). ومن هنا تبرز الصعوبة البالغة فى تحدث الإنسان مع الكمبيوتر.

ولحل هذه المشكلة ظهرت لغات البرمجة التى تعتبر بمثابة مترجم بين الإنسان والكمبيوتر. وبدلا من أن يتعلم الإنسان لغة الكمبيوتر الأصلية (تعرف أيضا بلغة الآلة machine language)، تستطيع تعلم لغة برمجة لتلقين الكمبيوتر بالتعليمات بطريقة سهلة.

ومن بين الأدوات المهمة التى تستخدم فى هذه العملية، برنامج متخصص يعرف

باسم المجمع compiler، ووظيفته هى تحويل التعليمات المكتوبة بلغة البرمجة إلى لغة الآلة. وهذا يعنى أنه يجب أن ألا تهتم كثيرا – أو حتى إطلاقا – بكيفية عمل الكمبيوتر، لكن يجب عليك التركيز على كيفية إمداده بالتعليمات من خلال لغة البرمجة.

تشبه لغة Visual Basic اللغة التى نستخدمها فى حياتنا اليومية. فعندما نتحدث أو نكتب شيئا ما، نستخدم أنواعا مختلفة من الكلمات، كالأسماء والأفعال. تحتوى لغة VB أيضا على أنواع مختلفة من الكلمات، تشكل عناصر اللغة. تشتمل هذه العناصر على الجمل البرمجية statements، الإعلانات declarations، الطرق methods، المعاملات operators، والكلمات المفتاحية keywords. وعندما تنتهى ـ عزيزى القارئ ـ من هذا الفصل، سوف تكون قادرا ـ بإذن الله ـ على فهم كل هذه العناصر، وكيفية استخدامها فى البرامج.

وعلى المنوال ذاته، نجد أن اللغات المنطوقة والمكتوبة تستخدم قواعد أو syntax يحدد ترتيب الكلمات فى الجملة. تتبع لغة VB قواعد معينة، قد تبدو غريبة فى البداية، لكنها ـ فى الواقع ـ بسيطة جدا. على سبيل المثال: لكتابة هذه الجملة (the maximum speed of my car is 80) بلغة VB، سوف تكتب الكود التالى:

```
Car.Speed.Maximum = 80
```

سوف تتعرف على قواعد كتابة الكود لاحقا، لكن يجب أن تعرف الآن أن هناك بعض الأدوات المهمة فى لغة VB – مثل IntelliSense– التى تساعدك على كتابة الكود أثناء إعداد البرنامج.

أضف إلى ذلك أن اللغة التى نقرأها ونكتبها تكون ذات بناء structure معين. على سبيل المثال: يتكون الكتاب من فصول، وتتكون الفصول من فقرات، وتحتوى الفقرات على جمل. كذلك الحال بالنسبة إلى البرامج التى نكتبها بلغة VB:

فالوحدات البرمجية Modules تشبه الفصول، والإجراءات Procedures تشبه الفقرات، وأسطر الكود تشبه الجمل.

دعنا الآن ـ عزيزى القارئ ـ نكتفى بهذا القدر من الأسس النظرية، ونبدأ التعرف عن قرب على أساسيات البرمجة العملية بلغة VB.

٤ـ٢ المتغيرات Variables

تعتبر المتغيرات من المفاهيم المهمة فى برمجة الحاسبات. والمتغير، ببساطة، هو مكان فى ذاكرة الكمبيوتر يحتوى قيمة ما، وله اسم معين يستخدم عند الرغبة فى تعيين/ استرجاع هذه القيمة. هذه القيمة قد تكون رقما مثل سعر الكتاب، أو اسما مثل اسم المؤلف.

هناك ثلاث خطوات لاستخدام المتغيرات:

١. الإعلان عن المتغير: من خلال تحديد اسمه ونوع البيانات التى يحتويها.

٢. تعيين قيمة للمتغير.

٣. استخدام المتغير: من خلال استرجاع القيمة المخزنة فيه.

٤ـ٢ـ١ الإعلان عن المتغير

يتم الإعلان عن المتغير باستخدام جملة Dim VariableName As DataType.

```
Dim aNumber As Integer
```

فى الكود السابق، قمنا بتحديد متغير اسمه aNumber وسوف نستخدمه فى تخزين بيانات من نوع الأرقام الصحيحة Integers.

إلى جانب الأرقام الصحيحة Integers كنوع بيانات، هناك أيضا البيانات من نوع Double وهى الأرقام ذات العلامة العشرية. كذلك هناك البيانات من نوع النصوص Strings كعنوان الكتاب مثلا. هذا بالإضافة إلى النوع Boolean الذى يستخدم لتخزين قيم مثل True or False.

٤-٢-٢ تعيين قيم للمتغيرات

يتم تخصيص قيمة للمتغير عن طريق علامة التساوى، كما فى المثال التالى:

```
aNumber = 42
```

الكود السابق يقوم بتخزين القيمة ٤٢ وهى رقم صحيح Integer فى المتغير aNumber.

ملحوظة مهمة

يمكن الإعلان عن متغير وتعيين قيمة له فى خطوة واحدة، كما فى المثال التالى:

```
Dim aDouble As Double = 3.14
Dim aName as String = "Ali Shaker"
Dim YesOrNo as Boolean =  True
```

تدريب عملى

فى هذا التدريب العملى سوف نقوم ببناء برنامج بسيط يقوم بتكوين أربعة متغيرات، وتخصيص قيم لهم، وعرض كل قيمة فى صندوق رسالة message box.

١. افتح برنامج لغة VB.NET 2005 Express Edition
٢. من قائمة File اختر New Project
٣. اختر Windows Application وأعط للبرنامج اسم Variables
٤. اضغط مرتين بالفأرة على سطح النموذج؛ وذلك لفتح نافذة الكود.
٥. اكتب الكود التالى فى الإجراء الخاص Form1.Load

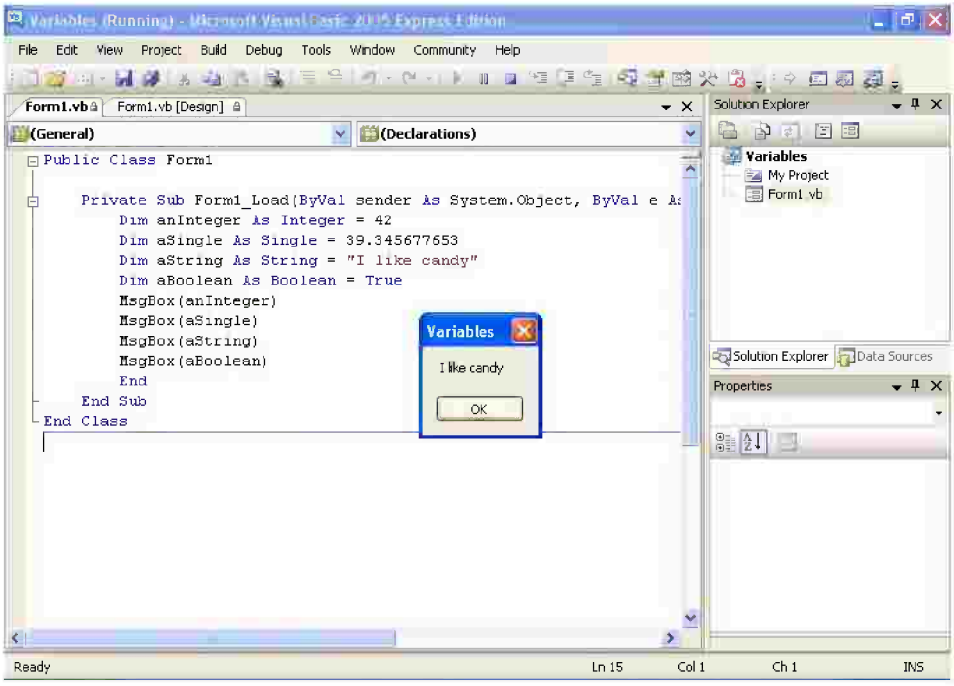
```
Dim anInteger As Integer = 42
Dim aSingle As Single = 39.345677653
Dim aString As String = "I like candy"
Dim aBoolean As Boolean = True
```

٦. وبعد ذلك اكتب الكود التالى:

```
MsgBox(anInteger)
MsgBox(aSingle)
MsgBox(aString)
MsgBox(aBoolean)
End
```

وظيفة الجزء الأخير من الكود هى عرض قيمة كل متغير فى صندوق رسالة عند تحميل النموذج على شاشة الكمبيوتر، ثم الانتهاء من البرنامج End.

٧. اضغط F5 لتشغيل البرنامج، مع الضغط على Ok فى كل رسالة؛ حتى ينتهى البرنامج.



شكل (١٤) استخدام المتغيرات Variables لتخزين البيانات

٤ـ٣ أنواع البيانات Data Types

تحدد أنواع البيانات فى لغة Visual Basic نوع القيم/البيانات التى يمكن تخزينها فى المتغير. ولعلك تتساءل عزيزى القارئ: لماذا توجد أنواع مختلفة من البيانات؟ للإجابة عن هذا التساؤل دعنا نتخيل أن هناك ثلاثة متغيرات، اثنين منهما يحتويان أرقاما، والثالث يحتوى على اسم. يمكنك تنفيذ عمليات حسابية على المتغيرين الأولين، لكنك لا تستطيع القيام بذلك مع الاسم. ومن هنا نستطيع أن نقول إن تحديد نوع البيانات للمتغير يسهل القيام بعمليات معينة.

ملحوظة

تستخدم أنواع البيانات فى – إلى جانب المتغيرات – العناصر الأخرى للغة البرمجة، مثل الثوابت constants، والخصائص properties، والدوال functions. وسوف نشرح ذلك فى جزء لاحق إن شاء الله.

٤ـ٣ـ١ أنواع البيانات الرقمية

تتعامل أغلب برامج الكمبيوتر مع الأرقام بشكل أو بآخر. ونظرا إلى أن هناك طرقا متعددة ومختلفة لتمثيل الأرقام، فإن Visual Basic تخصص أنواع متعددة للبيانات الرقمية؛ حتى يتعامل بكفاية مع الأرقام.

أكثر أنواع البيانات الرقمية استخداما هو الأرقام الصحيحة **Integers**، الذى يستخدم لتمثيل الأرقام الصحيحة (الرقم الصحيح هو الرقم الذى لا يحتوى علامة عشرية). عند اختيار نوع البيانات لتمثيل أرقام صحيحة، سوف تحتاج إلى نوع البيانات **Long** إذا كان المتغير سيحتوى على أرقام أكبر من مليارين تقريبا، وإلا اعتبر **Integer** مناسبا تماما لأغلب الحالات.

لكن ليست الأرقام كلها أرقاما صحيحة؛ على سبيل المثال، عند قسمة رقمين صحيحين، غالبا ما تكون النتيجة رقما يحتوى علامة عشرية. (٩ قسمة ٢ يساوى ٤.٥). وهنا يستخدم نوع البيانات **Double** لتمثيل الأرقام ذات العلامة العشرية.

ملحوظة

هناك أنواع أخرى للبيانات الرقمية، مثل **Decimal**، و **Short**، و **SByte**، و **UInteger** لكنها تستخدم فى البرامج الكبيرة جدا، حيثما تعتبر مساحة التخزين فى القرص الصلب مسألة مهمة. والأنواع التى قدمناها هنا تناسب أغلب التطبيقات. للتعرف على الأنواع الأخرى يمكنك ـ عزيزى القارئ ـ الرجوع إلى المراجع المذكورة فى نهاية الكتاب.

٤ـ٣ـ٢ أنواع البيانات النصية

تتعامل أغلب البرامج أيضا مع النصوص text، سواء عند عرض المعلومات للمستخدم، أو عند الحصول على مدخلات نصية من جانب المستخدم. يتم تمثيل النصوص غالبا باستخدام نوع البيانات **String**، الذى يمكنه أن يحتوى على سلسلة من الحروف والأرقام والمسافات والعلامات الخاصة. يمكن للبيان من نوع **String** أن يتفاوت فى الطول، فقد يكون جملة أو فقرة، أو حرفا واحدا، أو لاشيء على الإطلاق null string.

وبالنسبة إلى المتغير الذى يحتوى دائما على حرف واحد فقط، يمكن أيضا استخدام نوع البيانات **Char**. فإذا كنت تحتاج فقط إلى تخزين حرف واحد فى المتغير، تستطيع استخدام النوع **Char** بدلا من **String**.

٤ـ٣ـ٣ أنواع البيانات الأخرى

إلى جانب النصوص والأرقام، تحتاج البرامج أحيانا إلى تخزين أنواع أخرى من البيانات، مثل قيمة True أو False، أو تاريخ date، أو بيانات ذات معنى معين فى البرنامج.

بالنسبة إلى البيانات التى يتم تمثيلها على شكل true/false، أو yes/no، أو on/off، يخصص Visual Basic نوع البيانات Boolean، فهذا النوع من البيانات يمكنه تخزين إحدى قيمتين: True أو False.

وعلى الرغم من أنه يمكنك تمثيل التواريخ والأوقات كأرقام، فإن نوع البيانات **Date** يسهل عملية حساب التواريخ والأوقات، مثل عدد الأيام مذ يوم ميلادك، أو عدد الدقائق المتبقية حتى موعد الغذاء.

وعند الحاجة إلى تخزين أكثر من نوع واحد من أنواع البيانات فى متغير واحد، يمكن استخدام نوع البيانات composite. وتشتمل أنواع البيانات composite على arrays، و structures، و classes. وسوف تتعرف على المزيد حول هذه الأنواع فى أجزاء لاحقة من هذا الكتاب.

وأخيرا هناك بعض الحالات التى قد يختلف فيها نوع البيانات التى تريد تخزينها من وقت إلى آخر. وفى مثل هذه الحالة، يسمح نوع البيانات **object** بالإعلان عن متغير، ثم تحديد نوع بياناته لاحقا. لا داعى للقلق عزيزى القارئ، فسوف تتعرف على هذا النوع من بيانات فى درس لاحق بإذن الله.

٤.٤ المتغيرات النصية String

فى هذا الجزء سوف نتعرف معا ـ عزيزى القارئ ـ على نوع البيانات String ، والذى يستخدم لتمثيل الكلمات والنصوص.

فى الجزء السابق رأينا كيفية استخدام المتغيرات variables لتخزين بيانات يحتاجها البرنامج، كما رأينا كيف يجب أن يتوافق المتغير مع نوع البيانات الذى يختزنها. وفى هذا الدرس سوف نتعرف أكثر على نوع البيانات String، والذى يستخدم للتعامل مع النصوص.

٤.٤.١ ما تعريف String؟

يشير مصطلح String إلى أى سلسلة من الحروف النصية، كالحروف الهجائية والأرقام والعلامات الخاصة، بل حتى المسافات spaces.

يتم تكوين المتغيرات من نوع String بالطريقة ذاتها التى تتكون بها المتغيرات

عامة؛ حيث يجب الإعلان أولا عن المتغير، ثم تخصيص قيمة له، كما يتضح من المثال التالى:

```
Dim aString As String = "This is a string"
```

لعلك لاحظت ـ عزيزى القارئ ـ أن البيانات String يجب وضعها بين علامتى التنصيص (" "). كما يمكنك أيضا تخصيص متغير String إلى متغير آخر String عن طريق علامة التساوى. انظر المثال التالى:

```
Dim aString As String = "This is a string"
Dim bString As String = ""
bString = aString
```

يقوم الكود السابق بضبط قيمة المتغير bString؛ بحيث تساوى قيمة المتغير aString (وهى This is a string).

هذا فضلا عن أنه يمكنك استخدام علامة الواوية & للجمع بين المتغيرات من نوع String فى متغير جديد، كما يتضح من المثال التالى:

```
Dim aString As String = "Across the Wide"
Dim bString As String = "Missouri"
Dim cString As String = ""
cString = aString & bString
```

فى المثال السابق، أعلنا عن ثلاثة متغيرات من نوع String ، ثم قمنا بتخصيص القيمتين "Across the Wide" و "Missouri" لأول متغيرين. وبعد ذلك خصصنا حاصل جمعهما معا للمتغير الثالث. ولكن ما القيمة النهائية لعملية الجمع هذه؟ فى الواقع تكون القيمة Across the WideMissouri من دون مسافة فاصلة بين كلمتى Wide و Missouri . فإذا أردت هذه المسافة، فعليك كتابة الكود بالشكل التالى:

```
Dim aString As String = "Across the Wide"
```

```
Dim bString As String = "Missouri"
Dim cString As String = ""
cString = aString & " " & bString
```

الآن فقط تكون قيمة المتغير الجديد: Across the Wide Missouri.

تدريب عملي

الهدف من هذا التدريب هو ربط متغيرات من نوع String.

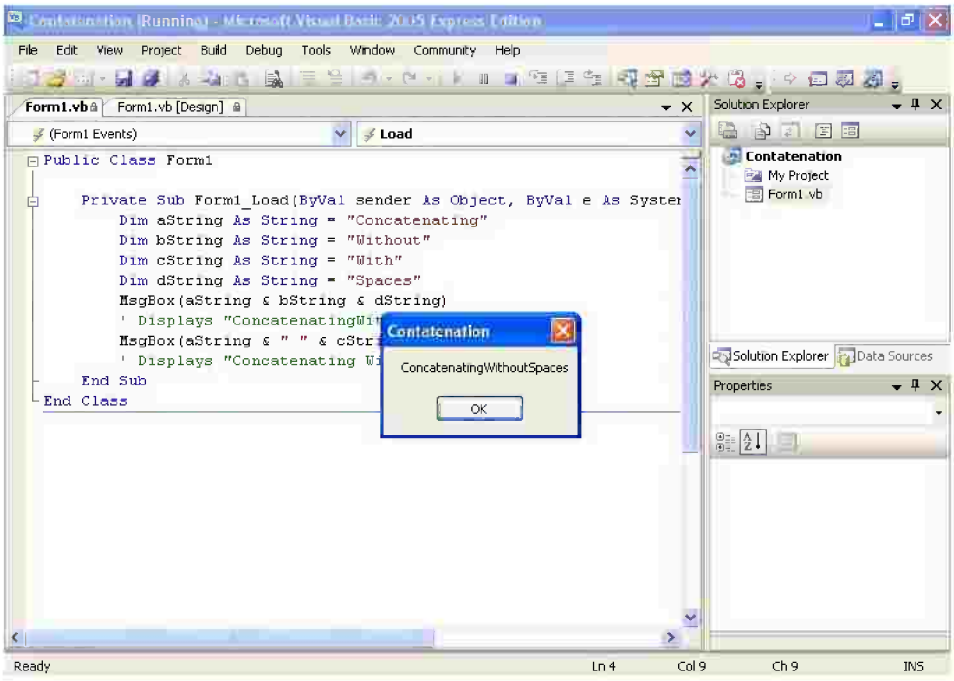
١. من قائمة File اختر New Project
٢. اختر Windows Application ثم أعط البرنامج اسم Concatenation.
٣. اضغط مرتين على سطح النموذج لفتح نافذة الكود.
٤. فى إجراء الحدث Form1.Load قم بالإعلان عن أربعة متغيرات من نوع String وحدد لهم القيم على النحو التالى:

```
Dim aString As String = "Concatenating"
Dim bString As String = "Without"
Dim cString As String = "With"
Dim dString As String = "Spaces"
```

٥. اكتب التالى للجمع بين قيم المتغيرات، واعرض النتائج:

```
MsgBox(aString & bString & dString)
' Displays "ConcatenatingWithoutSpaces"
MsgBox(aString & " " & cString & " " & dString)
' Displays "Concatenating With Spaces"
```

النص المعروض فى صندوق الرسالة هو نتيجة الجمع بين المتغيرات النصية. لكن الصندوق الأول يعرض النص من دون مسافات، على العكس من الصندوق الثانى.



شكل (٤-٢) جمع / ربط المتغيرات النصية Strings

٤-٥ المصفوفات Arrays

فى هذا الجزء سوف نتعرف معا ـ عزيزى القارئ ـ على كيفية استخدام المصفوفات؛ لتخزين مجموعة من القيم فى المتغير.

كما تعلمنا فى الأجزاء السابقة، كيف تستخدم المتغيرات لتخزين أنواع مختلفة من البيانات التى يحتاجها البرنامج. وهناك نوع آخر من المتغيرات، يعرف باسم المصفوفة Array، وهو يقدم طريقة سلسة لتخزين قيم متعددة من النوع ذاته فى المتغير الواحد.

على سبيل المثال: افترض أنك بصدد إعداد برنامج لفريق كرة قدم، ويريد تخزين أسماء جميع اللاعبين فى الفريق. فى هذه الحالة قد تستخدم ١١ متغيرا مستقلا، متغيرا لكل لاعب، أو يمكنك الإعلان عن مصفوفة، والتى تظهر فى الكود التالى:

```
Dim players( ) As String
```

٤-٥-١ الإعلان عن المصفوفة

يتم الإعلان عن المصفوفة باستخدام الهلاليتين بعد اسم المتغير، وإذا كنت تعرف عدد القيم التى تريد تخزينها، يمكنك تحديدها وقت الإعلان، على النحو التالى:

```
Dim players(10) As String
```

قد تتعجب ـ عزيزى القارئ ـ أن حجم المصفوفة ١٠، فى حين أن عدد لاعبى فريق كرة القدم ١١. والسبب فى ذلك أن عناصر المصفوفة تبدأ برقم صفر، وعلى ذلك تكون العناصر من صفر إلى ١٠، أى ١١ عنصرا.

٤-٥-٢ تعيين القيم فى المصفوفات

يتم تعيين قيمة لعنصر ما فى المصفوفة عن طريق تحديد رقم العنصر بين عناصر المصفوفة، ثم تحديد القيمة بعد علامة التساوى. انظر الكود التالى:

```
players(0) = "Hadi"
players(3) = "Emad"
```

فى المثال أعلاه قمنا بتخصيص القيمة Hadi للعنصر الأول فى المصفوفة (العنصر صفر)، كما قمنا بتخصيص القيمة Emad للعنصر الرابع فى المصفوفة. وجدير بالذكر هنا أنه لا يشترط تخصيص عناصر المصفوفة بالترتيب، كما أنه يمكن عدم تخصيص قيمة معينة لعنصر فى المصفوفة، وهنا سوف تكون قيمته فارغة.

بالإضافة إلى ذلك، يمكن الإعلان عن المصفوفة، وتخصيص قيم لها فى سطر واحد، تماما كما هى الحال بالنسبة إلى المتغيرات عموما.

```
Dim players() As Integer = {1, 2, 3, 4, 5, 6, 7, 8, 9,10,11}
```

فى هذه الحالة، تشير الأقواس المتعرجة curly braces إلى قائمة القيم. ولاحظ أن حجم المصفوفة غير محدد، لكن يمكن معرفته من خلال عدد العناصر الموجودة فى القائمة.

٤ـ٥ـ٣ استرجاع القيم من المصفوفة

يتم استخدام رقم العنصر فى المصفوفة لاسترجاع قيمته. انظر المثال التالى:

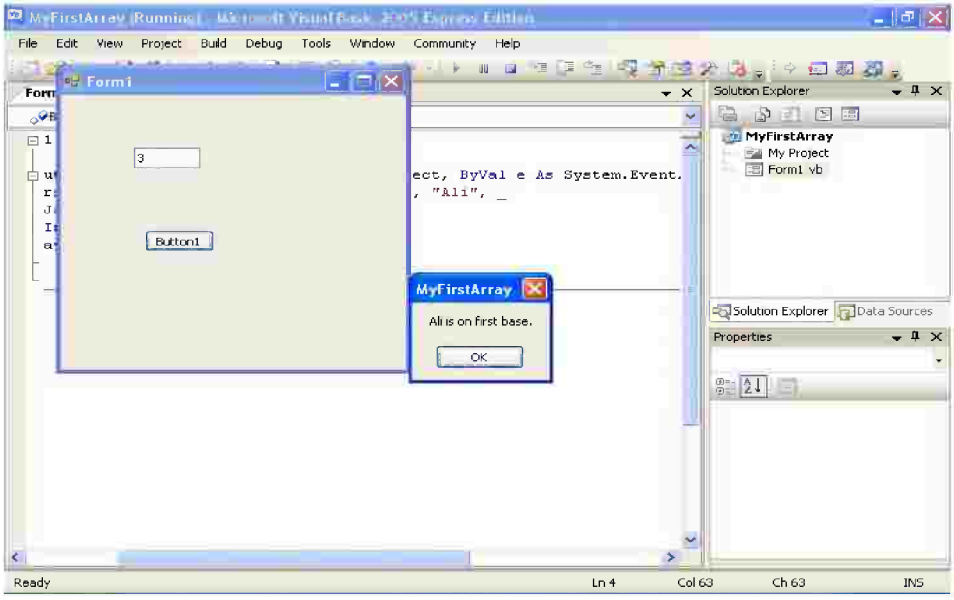
```
Dim AtBat As String
AtBat = players(3)
```

الكود السابق يسترجع العنصر الرابع فى المصفوفة، ويعين قيمته للمتغير AtBat.

تدريب عملى

الهدف من هذا التدريب هو تخزين قيم فى مصفوفة.

١. من قائمة File اختر New Project
٢. اختر Windows Application ثم أعط البرنامج اسم MyFirstArray.
٣. من صندوق الأدوات Toolbox، ضع صندوق نص Textbox على سطح النموذج.
٤. من صندوق الأدوات Toolbox، ضع زر أمر Button على سطح النموذج.
٥. اضغط مرتين على زر الأمر لفتح نافذة الكود.



شكل (٤ـ٣) المصفوفات Arrays

٦. فى إجراء الحدث Button1.Click اكتب الكود التالى:

```
Dim players() As String = {"Hadi", "Emad", "Ahmed", "Ali", _
 "Mohamed", "Tarek", "Nour", "Khaled", "Said", "Esam",
"Ayman"}
Dim i As Integer = CInt(Textbox1.Text)
MsgBox(players(i) & " is on first base.")
```

لاحظ أننا فى الكود السابق، استخدمنا الدالة Cint لتحويل القيمة النصية String الموجودة فى صندوق النص إلى رقم صحيح Integer (i).

٧. اضغط F5 لتشغيل البرنامج.

٨. اكتب رقما بين صفر و١٠ فى صندوق النص، ثم اضغط على زر الأمر، فسوف يظهر الاسم المخزن فى هذا العنصر فى صندوق الرسالة.

٦ـ٤ التحويل بين أنواع البيانات

رأينا فى جزء سابق، كيف أن هناك أنواعا مختلفة من المتغيرات، كل منها يحدد البيانات التى يمكن تخزينها. فالمتغير من نوع **Integer** يمكنه فقط تخزين البيانات الرقمية ذات الأعداد الصحيحة. أما المتغير من نوع **String** فيتعامل فقط مع النصوص.

لكن هل تساءلت ـ عزيزى القارئ ـ ما الذى يحدث عند الرغبة فى عرض رقم صحيح Integer فى صندوق نص String؟ الإجابة هى أن البيانات يجب تحويلها من نوع إلى آخر. وهذا هو بالضبط موضوعنا فى هذا الجزء. وفيه سوف نتعرف معا على الأساليب المستخدمة فى تحويل البيانات.

٦ـ٤ـ١ تحويل المتغيرات إلى نصوص

يمكن تحويل فى Visual Basic أى يمكن تحويله إلى نص، وذلك عن طريق الدالة **CStr** وهى اختصار Convert to String. هذه الدالة – كما يشير اسمها – تسترجع البيانات فى صورة نصية.

تدريب عملى

١. افتح برنامج لغة VB.NET 2005 Express Edition
٢. من قائمة File اختر New Project
٣. اختر Windows Application وأعط للبرنامج اسم Conversion
٤. اضغط مرتين بالفأرة على سطح النموذج؛ وذلك لفتح نافذة الكود.
٥. اكتب الكود التالى فى الإجراء الخاص Form1.Load

```
Dim anInteger As Integer = 54

MsgBox(CStr(anInteger))
```

فى هذا المثال قمنا بالإعلان عن متغير من نوع Integer وأطلقنا عليه اسم anInteger. كما قمنا بتعيين القيمة 54 لهذا المتغير. وبعد ذلك قمنا بتحويل هذه القيمة إلى نص، مع عرضها فى صندوق رسالة بعد استدعاء الدالة CStr.

٦. اضغط F5 لتشغيل البرنامج، تلاحظ ظهور صندوق الرسالة وفيه 54.

دعنا ـ عزيزى القارئ ـ نجرب شيئا ما على سبيل الدعابة. فى محرر الكود، غير السطر MsgBox(CStr(anInteger)) إلى MsgBox(anInteger)، ثم اضغط F5 لتشغيل البرنامج. ما الذى حدث؟ لقد عرض البرنامج صندوق الرسالة، محتويا 54 من دون أى تغير عند تشغيل البرنامج فى المرة السابقة. وتفسير ذلك أن لغة Visual Basic قامت بالتحويل الأوتوماتيكى للبيان الرقمى إلى نص (لعرضه فى صندوق الرسالة). لكننا لا ننصح بالاعتماد على ذلك بصورة مطلقة، ومن ثم يجدر بك استخدام الدالة CStr عند تحويل بيانات رقمية إلى نص.

وبالإضافة إلى تحويل الأرقام الصحيحة Integer إلى نص String، تستطيع الدالة CStr تحويل أية بيانات رقمية أخرى، مثل Double أو Long، إلى نص. كما أنها تستخدم أيضا لتحويل البيانات من نوع Date و Boolean إلى نص.

٤-٦-٢ التحويل بين أنواع البيانات الرقمية

رأينا عند دراستنا للعمليات الحسابية Arithmetic أن نتيجة العملية لا يمكن التعبير عنها برقم صحيح Integer. وكما توجد دالة لتحويل البيانات الرقمية – أيا كان نوعها – إلى نصوص، هناك أيضا دوال لتحويل متغير رقمى إلى متغير رقمى آخر. على سبيل المثال، يمكنك استخدام الدالة CDbl (Convert to Double) فى عملية حسابية لاسترجاع رقم عشرى عند التعامل مع متغيرات Integer.

تدريب عملى

١. فى المثال السابق (تحويل 54 من Integer إلى نص String وعرضه فى رسالة) امسح الكود واكتب بدلا منه الكود التالى:

```
Dim A As Integer = 1
Dim B As Integer = 2
MsgBox(CDbl(A / B))
```

فى الكود أعلاه قمنا بالإعلان عن متغيرين، مع تخصيص القيم 2 ,1 ثم قمنا بتحويل نتيجة عملية القسمة A / B باستخدام الدالة CDbl وعرضها فى صندوق رسالة.

٢. اضغط F5 لتشغيل البرنامج، يظهر صندوق رسالة، ويعرض 0.5

توجد دوال أخرى للتحويل بين البيانات الرقمية. على سبيل المثال، إذا أردت جمع قيمتين من نوع Double، وأردت تقريب النتيجة إلى أقرب رقم صحيح، يمكنك استخدام الدالة CInt.

٤-٧ العمليات الحسابية Arithmetic Operations

فى هذا الجزء، سوف نتعرف معا ـ عزيزى القارئ ـ على كيفية تنفيذ العمليات الحسابية. والتعبير الحسابى، ببساطة، هو عبارة عن كود يقوم باسترجاع قيمة معينة

بعد تنفيذ العملية. على سبيل المثال، عملية الجمع البسيطة 4+5 عبارة عن تعبير حسابى يقوم باسترجاع القيمة 9. يتكون هذا التعبير من جزءين: المتعاملات operands وهى القيم التى يتم إجراء العملية الحسابية عليها(5 و 4)، والمعامل operator وهو الذى يحدد نوع العملية المطلوب تنفيذها (علامة الجمع +).

٤ـ٧ـ١ استخدام القيم الناتجة عن العمليات الحسابية

حتى يصبح التعبير الحسابى مفيدا، يجب استخدام القيمة الناتجة عنه. أحد أشهر طرق الاستخدام هنا هى تخصيص هذه القيمة لمتغير معين، كما فى المثال التالى:

```
Dim anInteger As Integer = 5 + 4
```

هنا أعلنا عن متغير من نوع Integer أى عدد صحيح، وأطلقنا عليه اسم anInteger ثم خصصنا له القيمة الناتجة عن عملية جمع الرقمين.

٤ـ٧ـ٢ المعاملات الحسابية Arithmetic Operators

تعتبر العمليات الحسابية من أكثر طرق استخدام التعبيرات. وتشمل هذه العمليات: الجمع addition، والطرح subtraction، والضرب multiplication، والقسمة division. يوضح الجدول التالى أشهر المعاملات المستخدمة فى التعبيرات الحسابية.

المعامل	الوصف	
+ الجمع	يسترجع حاصل جمع قيمتين	5+7
- الطرح	يسترجع الفرق بين قيمتين	12-7
* الضرب	يسترجع حاصل ضرب قيمتين	3*4
/ القسمة	يسترجع حاصل قسمة قيمتين	12/4

قد يؤثر نوع المتغير الذى تستخدمه عند تنفيذ العملية الحسابية على النتيجة. فغالبا ما تؤدى قسمة رقمين إلى استرجاع قيمة ليست عددا صحيحا. على سبيل المثال: قسمة ٣ على ٢ ينتج عنها ١.٥. فإذا قمت بتخصيص القيمة الناتجة عن هذا التعبير إلى متغير Integer، فسوف يتم تقريبها إلى أقرب عدد صحيح، وهو ٢. لذا يجب عليك ـ عزيزى القارئ ـ عند القسمة، أن تستخدم المتغيرات من نوع Double التى تسمح بالأرقام العشرية؛ وذلك لتخزين ناتج عملية القسمة.

ملحوظة

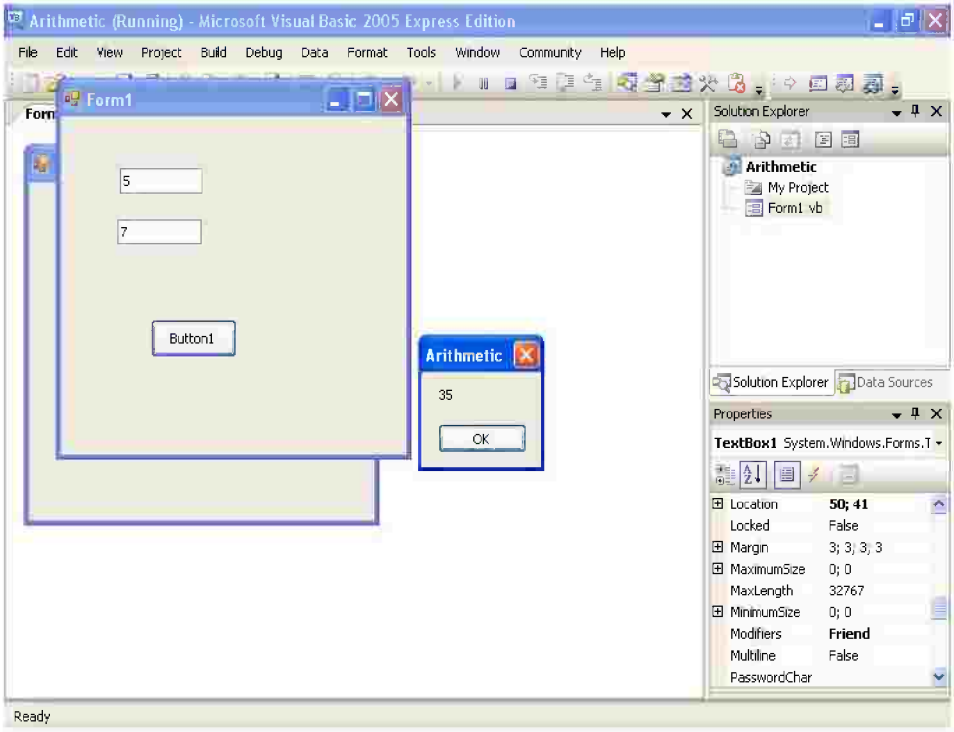
يمكنك تحويل متغير من نوع إلى نوع آخر باستخدام دالات التحويل conversion functions فى لغة Visual Basic. لمزيد من المعلومات: انظر الجزء الخاص بهذا الموضوع فى هذا الفصل.

تدريب عملي

١. من قائمة File اختر New Project
٢. اختر Windows Application ثم أعط البرنامج اسم Arithmetic.
٣. من صندوق الأدوات Toolbox، ضع صندوقى نص Textbox على سطح النموذج.
٤. من صندوق الأدوات Toolbox، ضع زر أمر Button على سطح النموذج.
٥. اضغط مرتين على زر الأمر لفتح نافذة الكود.
٦. فى إجراء الحدث Button1.Click اكتب الكود التالى:

```
Dim A As Double = Textbox1.Text
Dim B As Double = Textbox2.Text
MsgBox(A + B)
MsgBox(A - B)
MsgBox(A * B)
MsgBox(A / B)
```

قمنا بالإعلان عن المتغيرين A, B وذلك لتخزين القيم الحسابية المستخدمة فى البرنامج. كما قمنا بتخصيص القيم الموجودة فى صندوقى النص لهذين المتغيرين. وبعد ذلك قمنا ببناء أربعة تعبيرات حسابية، مع عرض نتيجة كل منهم فى صندوق رسالة.



شكل(٤.٤) تنفيذ العمليات الحسابية من خلال الكود

٧. اضغط F5 لتشغيل البرنامج.

٨. اكتب رقما معينا فى كل صندوق، ثم اضغط زر الأمر.

ملحوظة

إذا كتبت أية حروف أخرى غير الأرقام فى صندوق النص، فسوف يحدث خطأ فى تنفيذ البرنامج.

٤ـ٨ المقارنات Comparisons

فى هذا الجزء، سوف نتعرف معا ـ عزيزى القارئ ـ على استخدام معاملات المقارنات؛ وذلك لتكوين تعبيرات تقارن بين قيمتين.

ففى الجزء السابق، تعرفنا على المعاملات الحسابية. لكننا فى هذا الجزء نتعرف على نوع آخر من المعاملات، هو معاملات المقارنات comparison operators. وتستخدم هذه المعاملات لمقارنة قيم رقمية، وتسترجع قيم منطقية Boolean (صح True أو خطأ False).

كما تستخدم معاملات المقارنات فى عملية اتخاذ القرار بناء على نتيجة المقارنة. وسوف نقوم بشرح عملية اتخاذ القرارات فى كود البرنامج فى جزء لاحق من هذا الفصل.

يلخص الجدول التالى معاملات المقارنات:

المعامل	الوصف	أمثلة
= يساوى	تسترجع القيمة True إذا كان الرقم على اليسار (بالنسبة إلى اللغة الإنجليزية) يساوى الرقم على اليمين.	5=4 False 4=5 False 4=4 True
<> لا يساوى	تسترجع القيمة True عندما لا يتساوى الرقم على اليسار مع الرقم على اليمين.	5<>4 True 4<>5 True 4<>4 False
> أكبر من	تسترجع القيمة True إذا كان الرقم على اليسار أكبر من الرقم على اليمين.	5>4 True 4>5 False 4>4 False

> أقل من	تسترجع القيمة True إذا كان الرقم على اليسار أقل من الرقم على اليمين.	5<4 False 4<5 True 4<4 False
=< أكبر من أو يساوى	تسترجع القيمة True إذا كان الرقم على اليسار أكبر من أو يساوى الرقم على اليمين.	5>=4 True 4>=5 False 4>=4 True
=> أقل من أو يساوى	تسترجع القيمة True إذا كان الرقم على اليسار أقل من أو يساوى الرقم على اليمين.	5<=4 False 4<=5 True 4<=4 True

تدريب عملي

١. من قائمة File اختر New Project

٢. اختر Windows Application ثم أعط البرنامج اسم Comparison.

٣. من صندوق الأدوات Toolbox، ضع صندوق نص Textbox على سطح النموذج.

٤. من صندوق الأدوات Toolbox، ضع زر أمر Button على سطح النموذج.

٥. اضغط مرتين على زر الأمر لفتح نافذة الكود.

٦. فى إجراء الحدث Button1.Click اكتب الكود التالى:

```
Dim A As Double = CDbl(Textbox1.Text)
Dim B As Double = CDbl(Textbox2.Text)
MsgBox(A > B)
MsgBox(A < B)
MsgBox(A = B)
```

فى أول سطرين من الكود، قمنا بالإعلان عن المتغيرين A, B اللذين يستخدمان لتخزين القيم الرقمية فى البرنامج. وتلاحظ ـ عزيزى القارئ ـ أننا استخدمنا الدالة CDbl لتحويل النص المكتوب فى صندوق النص (من جانب المستخدم أثناء تشغيل البرنامج) إلى قيم رقمية. وبعد ذلك استخدمنا صندوق الرسالة لعرض نتائج المقارنات.

٧. اضغط F5 لتشغيل البرنامج.

٨. اكتب رقما فى كل صندوق من صندوقى النص، ثم اضغط زر الأمر.

فى أول صندوق رسالة سوف تظهر القيمة True إذا كانت A (الرقم الذى كتبته فى صندوق النص الأول) أكبر من B (الرقم المكتوب فى صندوق النص الثاني). وإذا كان الأمر غير ذلك، فسوف تظهر القيمة False فى صندوق الرسالة. وفى صندوق الرسالة الثانى، قسوف تظهر القيمة True إذا كانت A أقل من B. أما صندوق الرسالة الثالث فسوف تظهر فيه القيمة True إذا تساوت القيمتان.

ملخص

تعرفنا فى هذا الفصل ـ عزيزى القارئ ـ على أساسيات كتابة البرامج بلغة Visual Basic. والموضوعات التى ناقشناها هنا ـ عزيزى القارئ ـ لا تنطبق فقط على لغة VB وإنما هى من أساسيات البرمجة عموما، ولا ينبغى أبدا الإقلال من أهميتها. والمرجع الأساسى فى دراسة هذا الفصل يعود إلى حقيقة أن البرامج – صغيرة أو كبيرة – لا تستغنى أبدا عن البيانات Data. ومن ثم وجب على المبرمج معرفة كيفية تخزين هذه البيانات داخل متغيرات (لاسيما فى البرامج الصغيرة)، وأيضا معرفة أنواع هذه البيانات، وأسس الاختيار من بينها.

كما تعرضنا أيضا فى هذا الفصل إلى موضوع التحويل data conversion بين أنواع المتغيرات المختلفة. وأيضا كيفية تنفيذ العمليات الحسابية، والمقارنات المنطقية من خلال الكود.

تمارين

١. ما المتغيرات؟ ومتى تستخدم؟.

٢. يستخدم نوع البيانات ..... فى تخزين أى سلسلة من الحروف النصية، كالحروف الهجائية والأرقام والعلامات الخاصة.

٣. لتحويل المتغيرات النصية إلى متغيرات رقمية، تستخدم الدالة (اختر واحدة):

a. CStr

b. CDbl

c. CInt

٤. للإعلان عن مصفوفة من ١٠ عناصر، نكتب الكود التالى (صح أم خطأ):

Dim players(١٠) As String

٥. ضع دائرة حول نتيجة كل من المقارنات التالية:

a. 3 > 5 True False

b. 5 < 3 True False

c. 4 <= 6 True False

d. 4 >= 6 True False