Microsoft®
Visual Studio®2010

الوحدة السابعة

البرمجة بلغة (فيجول بيسك ستوديو)

موضوعات الوحدة :

- مراحل كتابة البرنامج بلغة (فيجول بيسك ستوديو).
- طريقة تعامل البرنامج مع البيانات.
- العمليات الحسابية والمنطقية.
- أدوات البرمجة بلغة (فيجول بيسك ستوديو).
- بعض الأوامر الأساسية للغة (فيجول بيسك ستوديو).

بعد دراستك لهذه الوحدة سوف تحقق الأهداف التالية :

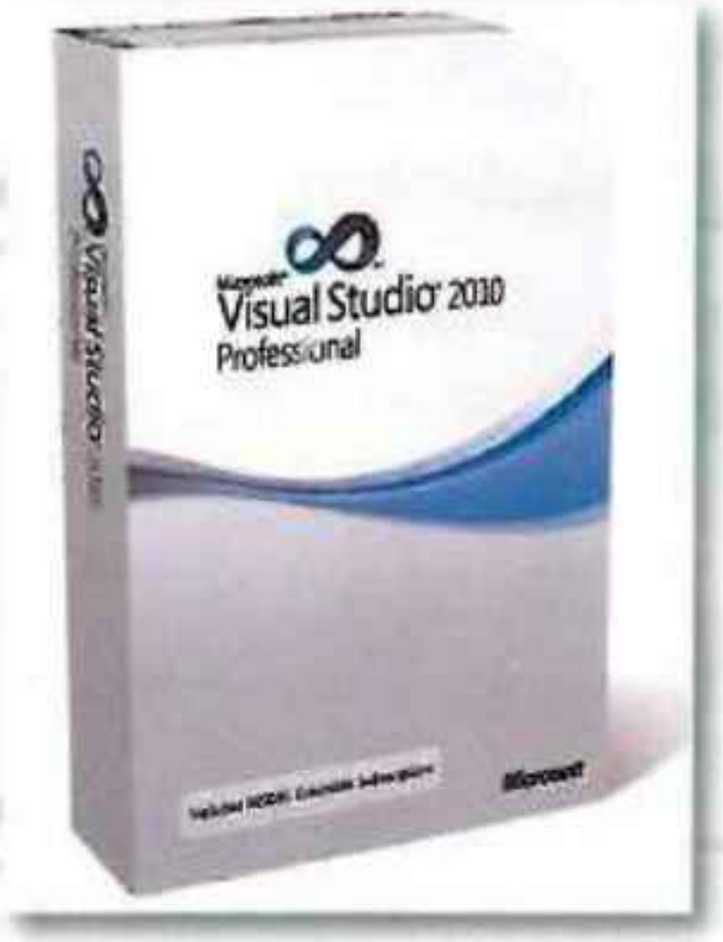
- تُعدِّد مراحل كتابة البرنامج بلغة فيجول بيسك ستديو.
- تُوضِّح طريقة تعامل برنامج فيجول بيسك ستديو مع البيانات.
- تُجري العمليات الحسابية والمنطقية ببرنامج الفيجول بيسك ستديو.
- تستخدم أدوات البرمجة بلغة فيجول بيسك ستديو.
- تُعدِّد الأوامر الأساسية في لغة فيجول بيسك ستديو.

الأهميـــة :

الهدف الرئيس لجهاز الحاسب هو القيام بالعمليات الحسابية ومعالجة البيانات وهذه المهمة تتطلب وجود برامج تقوم بها، هذه البرامج يقوم ببنائها المبرمجون ويحتاجون إلى منصات عمل توفر لهم أدوات قوية عند التنفيذ.

وتعد لغة فيجول بيسك ستديو من لغات المستوى العالي سهلة التعلم والتي غالبًا ما ينصح المبتدئين في عالم البرمجة بتعلمها وذلك لخلوها من التعقيد واعتمادها على البرمجة بالكائنات «البرمجة الشيئية» مع مناسبتها لتطبيقات قواعد بيانات والتطبيقات المخصصة للشركات الصغيرة.

١-٧ مقدمة



هناك عدة لغات برمجية لإنشاء برامج خاصة بالحاسب كما تعلمت سابقًا، وسوف ندرس في هذا الوحدة إحدى اللغات العالية (High-Level Language)، وبالتحديد إحدى لغات البرمجة بالعناصر أو البرمجة المرئية وهي لغة (فيجول بيسك ستوديو) (Visaul Basic Studio).

وتعد البرمجة باستخدام (فيجول بيسك ستوديو) شيقة وممتعة، وذلك لما تمتاز به من تحكم المبرمج في البرامج التي يقوم بتصميمها من ناحية: واجهات الإدخال للمستخدم، والعمليات الإجرائية للبرنامج، وأخيراً المخرجات التي يحصل عليها المستخدم لهذا البرنامج.

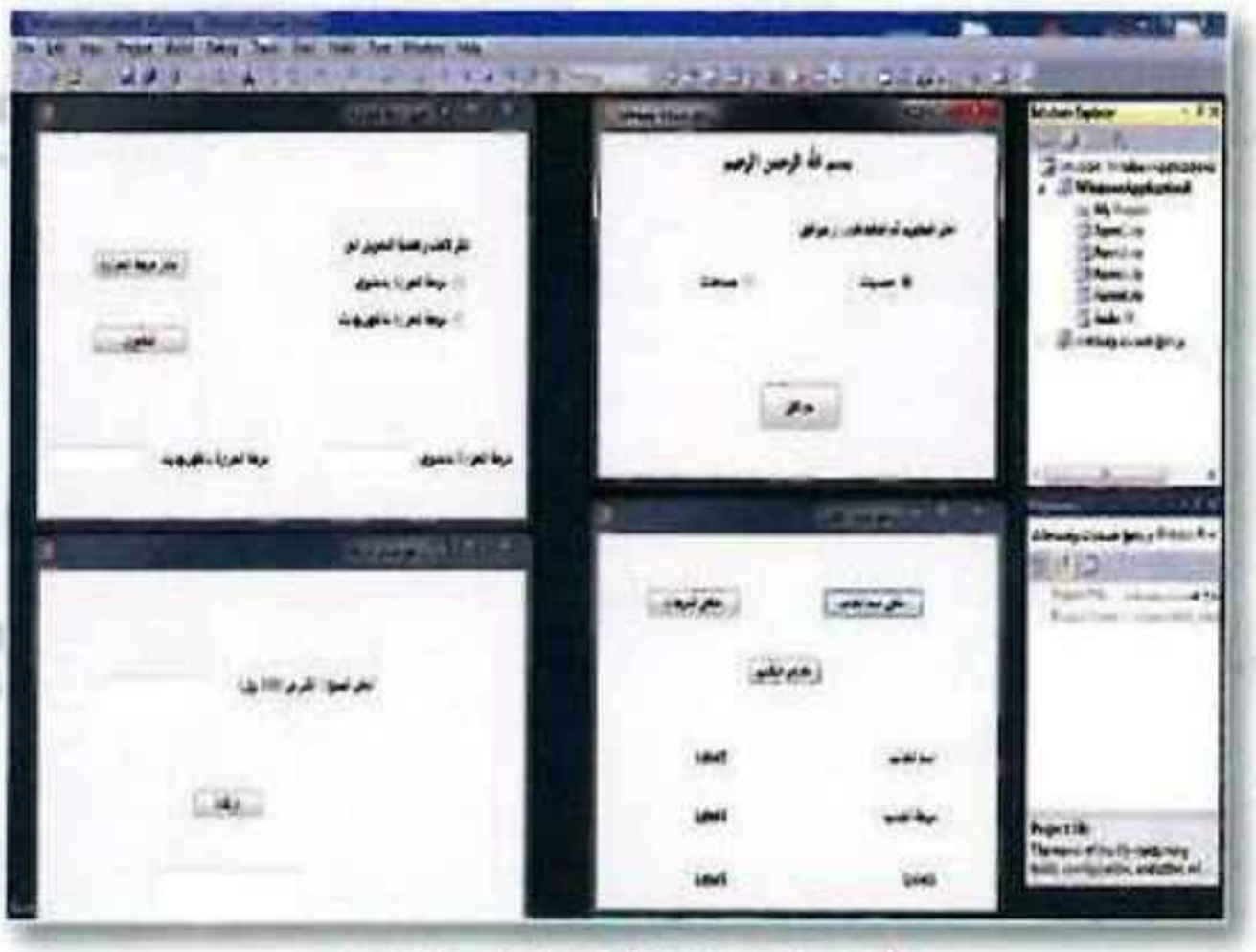
٢-٧ مراحل كتابة البرنامج بلغة (فيجول بيسك ستوديو)

تعلمنا في الوحدة السابقة خطوات المرحلة الأولى من حل المسألة وهي: فهم المسألة وتحديد عناصرها، وكتابة الخوارزم والخطوات المنطقية للحل، والتمثيل البياني للخوارزم عن طريق مخططات الانسياب. وفي هذه الوحدة ستتعلم المرحلة الثانية وهي مرحلة كتابة البرنامج باستخدام لغة (فيجول بيسك ستوديو)، والتي تتكون من ثلاث خطوات:

١. تصميم الواجهات.
٢. ضبط خصائص الأدوات.
٣. كتابة أوامر البرمجة.

أولاً تصميم الواجهات:

وهنا نبدأ تصميم الواجهات التي سوف تظهر للمستخدم، من: تحديد عدد النوافذ التي يحتاجها البرنامج، والأدوات التي نحتاجها على كل نافذة، كالأزرار ومربعات النصوص والقوائم، وغيرها كما في الشكل (٧-١).



شكل (٧-١): نماذج لواجهات برامج

ثانيا ضبط خصائص الأدوات :

بعد أن نضع الأدوات على النافذة، تأتي مرحلة تحديد خصائص هذه الأدوات، حيث توجد لكل أداة من الأدوات عدة خصائص (Properties) كشكلها، ولونها، والخط المستخدم فيها، وعنوانها، وغير ذلك. وهذه الخصائص افتراضية، لذا نقوم بتغيير خصائص هذه الأدوات لتناسب البرنامج، كما في الشكل (٧-٢).

نافذة خصائص الأدوات وفيها نحدد خصائص كل أداة .

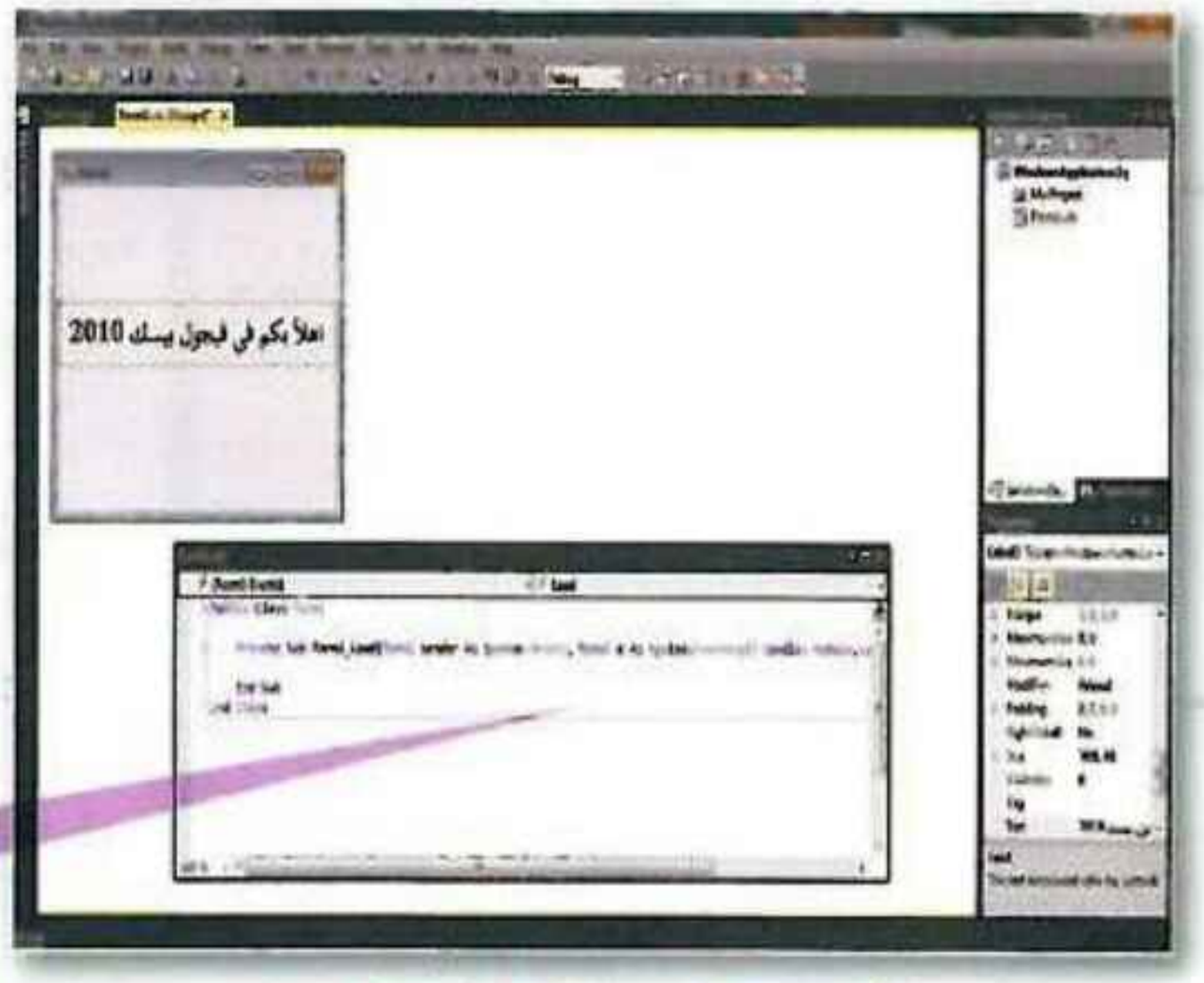
مثلاً لتغيير خاصية عنوان أداة تجد في نافذة الخصائص خاصية (Text) وهكذا لبقية الخصائص.

شكل (٧-٢) : نافذة البرنامج وخصائص الأدوات

ثالثا كتابة أوامر البرمجة :

بعد أن ننتهي من المرحلتين السابقتين، تأتي مرحلة كتابة الأوامر التي نريد من (فيجول بيسك ستوديو) أن ينفذها عند وقوع حدث معين، فمثلاً عندما يضغط المستخدم على زر الأوامر ففي هذه الحالة يكون الحدث، وهنا نكتب الأوامر التي نريد من البرنامج أن ينفذها، كما في الشكل (٧-٣).

شاشة كتابة الأوامر حيث يكون الاعلان عن المتغيرات التي تحتاجها وغيرها من الأوامر.



شكل (٧-٣) : شاشة كتابة أوامر البرمجة

٣-٧ طريقة تعامل البرنامج مع البيانات

الهدف الرئيس من أي برنامج هو معالجة البيانات. وتختلف أنواع البيانات، فقد تكون حروفًا أو أرقامًا أو تواريخ أو غيرها. وتأتي البيانات غالباً من مستخدم البرنامج، حيث يدخلها عن طريق أجهزة الإدخال المتصلة بجهاز الحاسب الآلي كلوحة المفاتيح أو الفأرة مثلاً.

يستقبل البرنامج البيانات ويخزنها في الذاكرة الرئيسة حتى يستطيع استخدامها والتعامل معها. لذلك لا بد من إعطاء هذه البيانات أسماء معينة ليتمكن من الرجوع إليها، وتصنف هذه البيانات إلى نوعين : ثوابت ومتغيرات.

أولا الثوابت وأنواعها :

نحتاج في بعض البرامج إلى التعامل مع بعض الثوابت باستمرار، وبدلاً من كتابة قيمة هذا الثابت في كل مرة نستخدمه يمكن إعطاء هذا الثابت اسمًا معينًا يستخدم بدلاً منه.

الثابت هو إعطاء اسم لقيمة معينة ويستخدم داخل البرنامج. ولا يمكن تغيير هذه القيمة عند تنفيذ البرنامج.

إثراء علمي

عندما نحتاج إلى تغيير قيمة الثابت، فإنه يمكن تغييره في المكان الذي تم تعريفه فيه بدلًا من تغيير هذه القيمة في كل مرة استخدمنا فيها الثابت في البرنامج.

أنواع الثوابت :

١) ثابت عددي.
٢) ثابت حرفي.

طريقة تعريف الثوابت :

تُعرَّف الثوابت باستخدام الأمر (Const)

```
Const Const1 = Value
```

حيث إن :

١) **Const**: الأمر الذي نستخدمه لتعريف الثابت.
٢) *Const1*: اسم الثابت، ويتم اختياره من قبل المبرمج .
٣) *Value*: القيمة التي سوف تخزن في هذا الثابت .

مثال

إذا أردنا تعريف ثابت الدائرة (النسبة بين محيط الدائرة وقطرها) يكون كالتالي :

```
Const Pi= 3.14
```

ثانيا المتغيرات وأنواعها :

المتغير هو مكان في الذاكرة الرئيسة تخزن فيه بيانات وتعطى اسمًا معينًا حتى يتم استرجاعها والتعامل معها داخل البرنامج، ويمكن تغيير ما يتم تخزينه، لذلك سميت بالمتغيرات. وتختلف المتغيرات باختلاف البيانات التي تخزن بها. كل متغير له اسم ونوع وقيمة.

أنواع المتغيرات :

تتعدد المتغيرات بحسب نوع البيانات التي تخزن بها، فمثلاً قد يكون رقمًا صحيحًا أو رقمًا عشريًا أو حرفًا أو مجموعة حروف. ويمكن تقسيم الأنواع إلى:

١. متغير عددي.
٢. متغير حرفي.
٣. متغير منطقي.

شروط تسمية المتغير :

ذكرنا أن المتغير يعطى اسمًا من قبل المبرمج، ولكنّ لهذا الاسم شروطًا معينة وهي:

١. أن يتكون من حروف انجليزية (A..Z)، وأرقام، والرمز (___)، ولا يحتوي على فراغ أو أي رمز آخر.
٢. أن لا يبدأ برقم.
٣. أن لا يتجاوز (٢٥٥) حرف.
٤. أن لا يكون محجوزًا للغة البرمجة.

أمثلة على أسماء صحيحة للمتغيرات:

X Y A23 C_d

أمثلة على أسماء غير صحيحة للمتغيرات:

DIM 2DF IF@

طريقة تعريف المتغير :

تُعرَّف المتغيرات في لغة (فيجول بيسك ستوديو) باستخدام الأمر (Dim) وصيغته كالتالي:

```
Dim Var1 As Type
```

حيث إن :

١ **Var1**: اسم المتغير.

٢ *As* : رابط بين اسم المتغير ونوعه (من الأسماء المحجوزة للغة فيجول بيسك ستوديو).

٣ *Type* : نوع المتغير.

ويمكن تعريف أكثر من متغير في الأمر نفسه : *Dim Var1 As Type, Var2 As Type, ...*

مثال

إذا أردنا تعريف متغير لتخزين اسم الطالب وليكن (name)، وهو من نوع متغير حرفي.

يكون كالتالي: **Dim *name* As String**

ولتعريف أكثر من متغير: **Dim *name* As String, *age* As Integer**

إثراء علمي

الأسماء المحجوزة (reserved words): هي الأسماء التي تكون إما أسماء لأنواع البيانات الموجودة في لغة البرمجة، أو أسماء لأوامر في اللغة لا يسمح باستخدامها. مثلاً في لغة (فيجول بيسك ستوديو) من الأمثلة على الأسماء المحجوزة الكلمات التالية:
DIM .IF.FOR

أنواع البيانات :

تتعامل لغة (فيجول بيسك ستوديو) مع أنواع مختلفة من البيانات، ولكل من هذه الأنواع اسم معين وسعة تخزينية معينة، نلخصها في الجدول التالي :

نوع البيانات	الاسم	الحجم	طريقة التعريف	مثال
عدد صحيح	Integer	٢بايت	Dim X As Integer	X=25
عدد صحيح طويل	Long	٤بايت	Dim Y AS Long	Y=12500000
عدد عشري	Single	٤بايت	Dim X2 As Single	X2=10.5
عدد عشري مضاعف	Double	٨بايت	Dim Y2 As Double	Y2=10.55555678

نوع البيانات	الاسم	الحجم	طريقة التعريف	مثال
العملة	Currency	٨بايت	Dim SR as Currency	SR=100.00
متسلسلة نصية	String	بايت لكل حرف	Dim UserName as String	UserName="Admin"
منطقي	Boolean	٢بايت	Dim B As Boolean	B=true
تاريخ	Date	٤بايت	Dim D As Date	D=#04-10-99#
متنوع (أي يمكن تخزين أي من الأنواع السابقة)	Variant	١٦بايت	Dim Var as Variant	Var=55.12 Var="Hello"

٤-٧ العمليات الحسابية والمنطقية

أولاً العمليات الحسابية في البرمجة :

تحتوي جميع لغات البرمجة على عمليات الحساب الأساسية : الجمع والطرح والضرب والقسمة والأس. وتختلف طريقة كتابة المعادلات الحسابية عن الطريقة الجبرية كما هو موضح في الجدول التالي:

العملية	الرمز	الصيغة الجبرية	الصيغة البرمجية
الجمع	+	$x + y$	$x + y$
الطرح	-	$x - y$	$x - y$
الضرب	*	xy	$x * y$
القسمة	/	$x \div y$ أو $\frac{x}{y}$	x / y
الأس	^	x^y	x^y

لاحظ الاختلاف في طريقة كتابة عمليتي الضرب والقسمة والأس.

قد تحتوي المعادلة الحسابية على أكثر من عملية مثال: $x + y / z$

عند محاولة حل هذه المعادلة يبرز لدينا سؤال مهم : هل تنفذ عملية الجمع أولاً أو عملية القسمة؟

بافتراض أن : $x=2، y=4، z=2$

جرّب تنفيذ عملية الجمع أولاً ثم عملية القسمة ثم اعكس الترتيب.

هل الناتج نفسه؟

.....

نخلص من هذا إلى أن الترتيب في تنفيذ العمليات يؤثر على الناتج، لذلك وجب أن تكون لدينا قوانين نتبعها لنعرف أي العمليات تنفذ أولاً.

ترتيب العمليات الحسابية :

١. العمليات التي في داخل الأقواس.
٢. عمليات الأس.
٣. عمليات الضرب والقسمة، وإذا تعددت نبدأ التنفيذ من اليسار إلى اليمين.
٤. عمليات الجمع والطرح، وإذا تعددت نبدأ التنفيذ من اليسار إلى اليمين.

مثال ١ :

ما نتيجة تنفيذ العملية التالية على جهاز الحاسب: $M= 2*6 / 3$ ؟

الحـل:

حيث إن العمليات هنا هي الضرب والقسمة ولها نفس الأولوية نفسها فسوف نبدأ التنفيذ من اليسار لليمين:

نتفذ عملية الضرب أولاً: $M=12 / 3$

ثم عملية القمسة ثانياً: $M=4$

مثال ٢ :

ما نتيجة تنفيذ العملية التالية على جهاز الحاسب: $M=2*6 +3\text{^}2$ ؟

الحـل:

حيث إن العمليات هنا هي الضرب والجمع والأس ولها أولويات مختلفة فسوف نبدأ التنفيذ بالترتيب:

نتفذ عملية الأس أولاً: $M = 2 * 6 + 9$

ثم عملية الضرب ثانياً: $M = 12 + 9$

وأخيراً عملية الجمع: $M=21$

مثال ٣: ما نتيجة تنفيذ العملية التالية على جهاز الحاسب: $M=2*(6+3)\wedge 2$ ؟

الحل:

حيث إن العمليات هنا هي الضرب والجمع والأس ولها أولويات مختلفة، لكن يوجد أقواس حول عملية الجمع لذلك نبدأ بها:

ننفذ عملية الجمع أولاً:	$M=2*9\wedge 2$
ثم عملية الأس ثانياً:	$M=2*81$
وأخيراً عملية الضرب:	$M=162$

ثانيا العمليات المنطقية في البرمجة :

ويقصد بها العمليات التي تتم فيها المقارنة بين قيمتين، سواء أكانتا عدديتين أو حرفيتين، متساويتين أو غير متساويتين، أو إحداهما أكبر أو أصغر من الأخرى. ويوضح الجدول التالي عمليات المقارنة المستخدمة في (فيجول بيسك ستوديو).

العامل	معناه
=	يساوي
<>	لا يساوي
>	أكبر من
<	أصغر من
>=	أكبر من أو يساوي
<=	أصغر من أو يساوي

يكون الناتج في عمليات المقارنة إما القيمة (True) أي : صحيح أو (False) أي : خطأ.

لو كان لدينا عمليات حسابية ومعها عملية مقارنة فإن أولوية التنفيذ تكون للعمليات الحسابية.

مثال ١: ما نتيجة تنفيذ العملية التالية على جهاز الحاسب: $4 <= 10$ ؟

الحل: النتيجة: (True) أي : صحيحة؛ لأن 10 فعلاً أكبر من 4

مثال ٢ :

ما نتيجة تنفيذ العملية التالية على جهاز الحاسب: $20 < 12$ ؟

الحل : النتيجة: (False) أي : خطأ؛ لأن 12 ليست أكبر من 20

مثال ٣ :

ما نتيجة تنفيذ العملية التالية على جهاز الحاسب: $6 * 4 > 5 * 3 + 4$ ؟

الحل : تنفذ العمليات الحسابية أولاً:

$4 + 15 < 24$

$19 < 24$

النتيجة: (True) أي : صحيحة؛ لأن 19 فعلاً أصغر من 24

ثالثاً تحويل المعادلات الجبرية إلى الصيغة المستخدمة في البرمجة :

لاحظنا عند دراسة العمليات الحسابية أن طريقة كتابتها بالصيغة الجبرية تختلف عن طريقة كتابتها بالصيغة البرمجية. وعند قيامك بخطوة صياغة حل المسألة فغالباً ما تكون العمليات الحسابية مكتوبة بالصيغة الجبرية؛ لذلك يجب عليك عند كتابة البرنامج تحويل العمليات الحسابية من الصيغة الجبرية إلى الصيغة البرمجية.

مثال ١ :

حوِّل المعادلة الجبرية الآتية إلى معادلة بصيغة برمجية.

$$Num = \frac{X^2}{A + B}$$

الحل : $Num= X \text{ ^ } 2/(A+B)$

مثال ٢ :

حوِّل المعادلة الجبرية الآتية إلى معادلة بصيغة برمجية.

$X = 5Y- 4\div1$

الحل : $X = 5*Y- 4/1$

٧-٥ أدوات البرمجة بلغة (فيجول بيسك ستوديو)

أولا أدوات البرمجة :

الأدوات(Tools) تعرف بأنها أجزاء برامج جاهزة للاستخدام، أي أنها أعدت مسبقًا لتوفّر على المبرمج **الوقت والجهد.** وتستخدم هذه الأدوات لإجراء عمليات الإدخال والإخراج، ويتم ربطها بأوامر البرمجة التي تعالج البيانات المدخلة لتخرج لنا المعلومات المطلوبة.

فكل ما يجب عليك عمله لاستخدام هذه الأدوات هو:

- وضعها على النموذج في المكان المناسب.
- ضبط الخصائص الخاصة بالأداة.
- كتابة أوامر البرمجة التي تتعامل مع هذه الأداة.

ثانيا خصائص الأدوات :

تختلف خصائص الأدوات باختلاف الأدوات، ولكن هناك خصائص مشتركة تشترك فيها كل الأدوات وهي التي سنتعرف عليها هنا. أما الخصائص الخاصة بكل أداة فسوف نتعرف عليها عند شرح كل أداة.

١ الخصائص المشتركة بين الأدوات :

الأداة	الخاصية
Name	تحديد اسم الأداة.
Textalign	تحديد محاذاة النص المكتوب (يمين- يسار-وسط).
Text	إظهار عنوان للنموذج أو نص داخل الأداة على الواجهة.
Font	تغيير نوع الخط وحجمه ونمطه.
ForeColor	تغيير اللون المكتوب به النص.
BackColor	تغيير لون الخلفية للأداة أو النموذج.
Location	تحديد موقع الأداة داخل النموذج.
Size	تغيير حجم النموذج أو الأداة.
Visible	إظهار أو إخفاء الأداة.

٢ ضبط خصائص الأدوات :

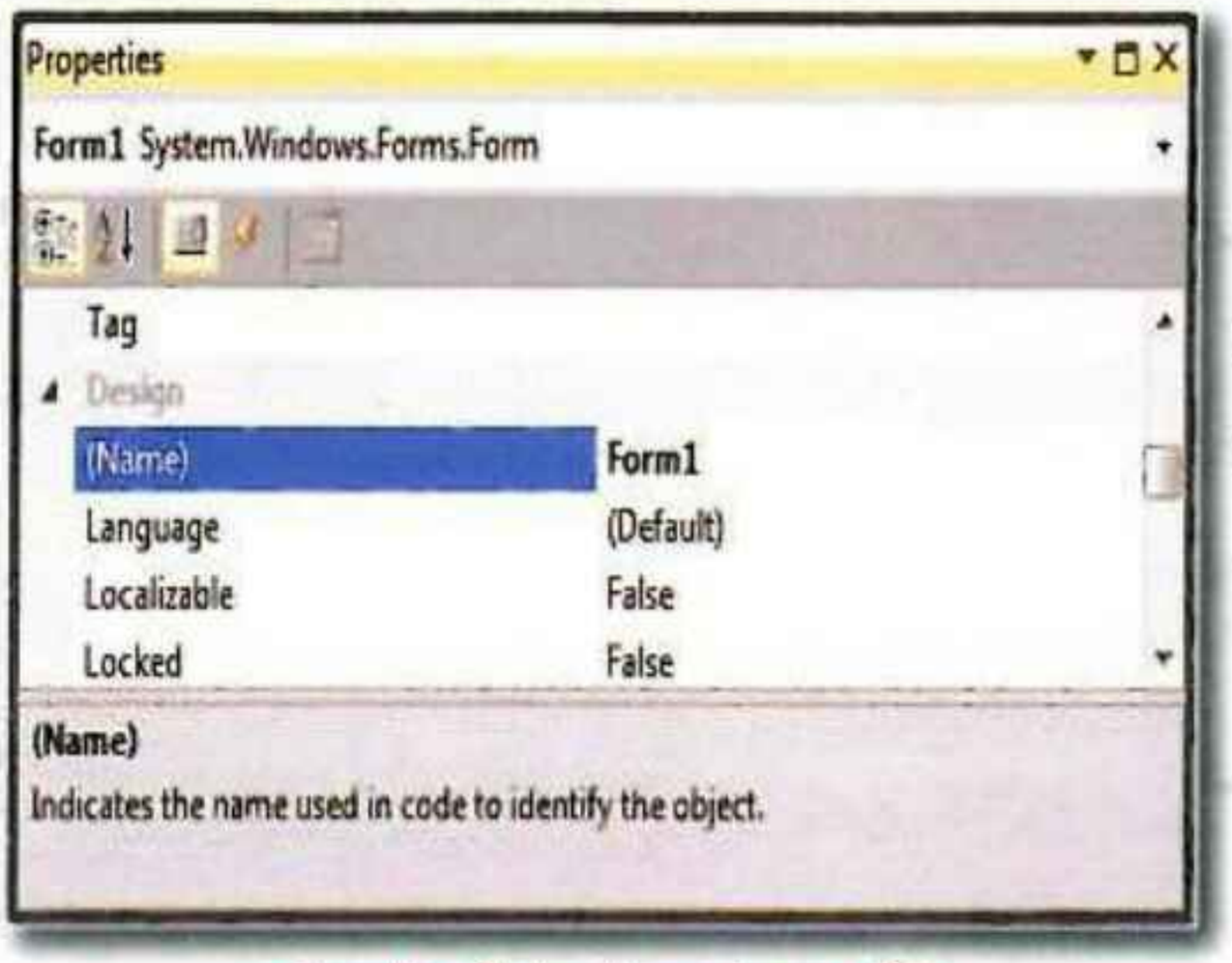
هناك طريقتان للتحكم بخصائص الأدوات، هما:

أ ضبط خصائص الأداوت أثناء تصميم البرنامج :

نستخدم إحدى الطرق التالية:

١ - كتابة القيمة :

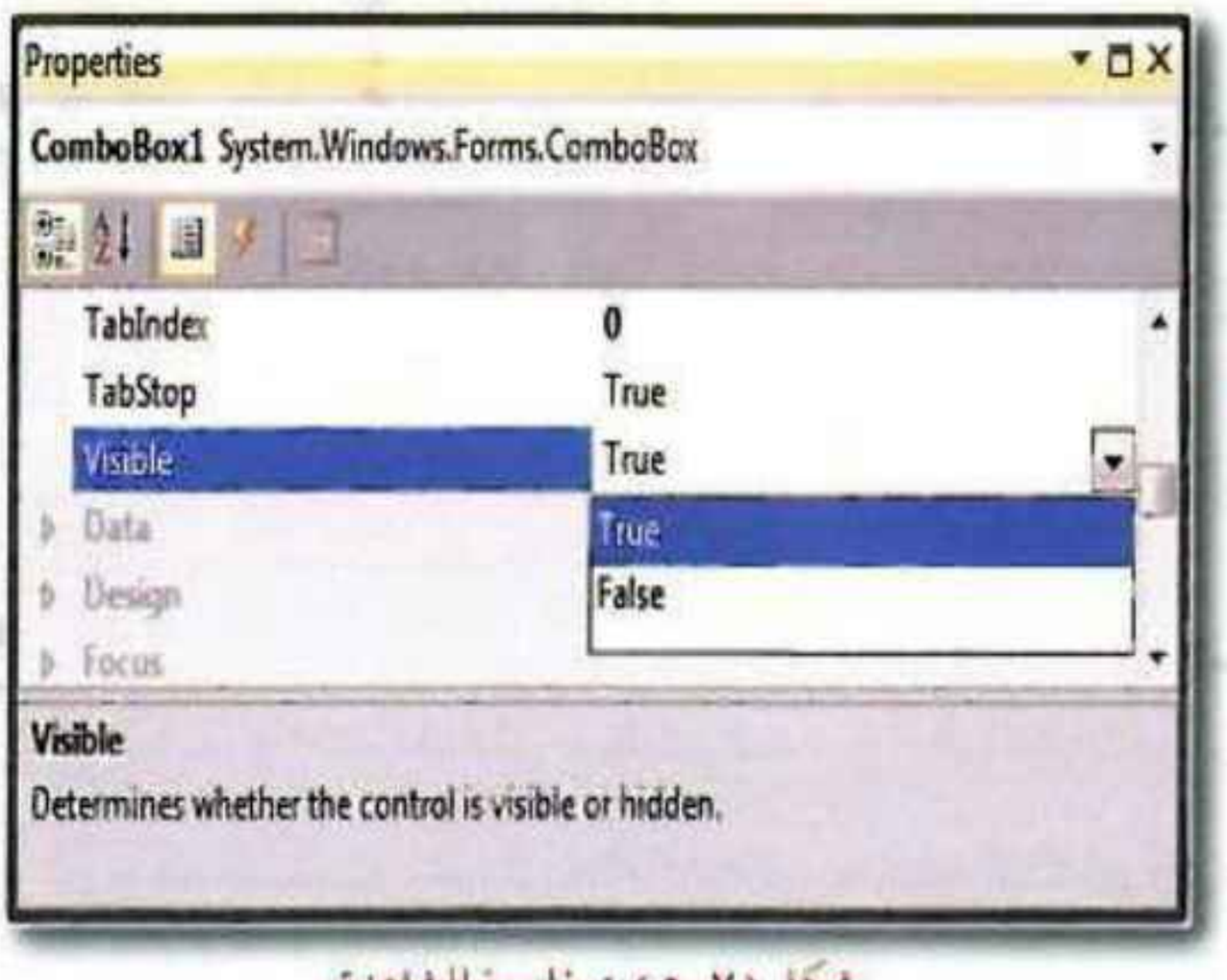
نكتب القيمة بواسطة لوحة المفاتيح في الخانة المخصصة داخل إطار الخصائص. كما في خاصية الاسم (Name) في الشكل (٧-٤)، حيث كتبنا القيمة (Form1) داخل خانة (الاسم).



شكل (٧-٤) : خاصية الاسم للنموذج

٢ - اختيار القيمة :

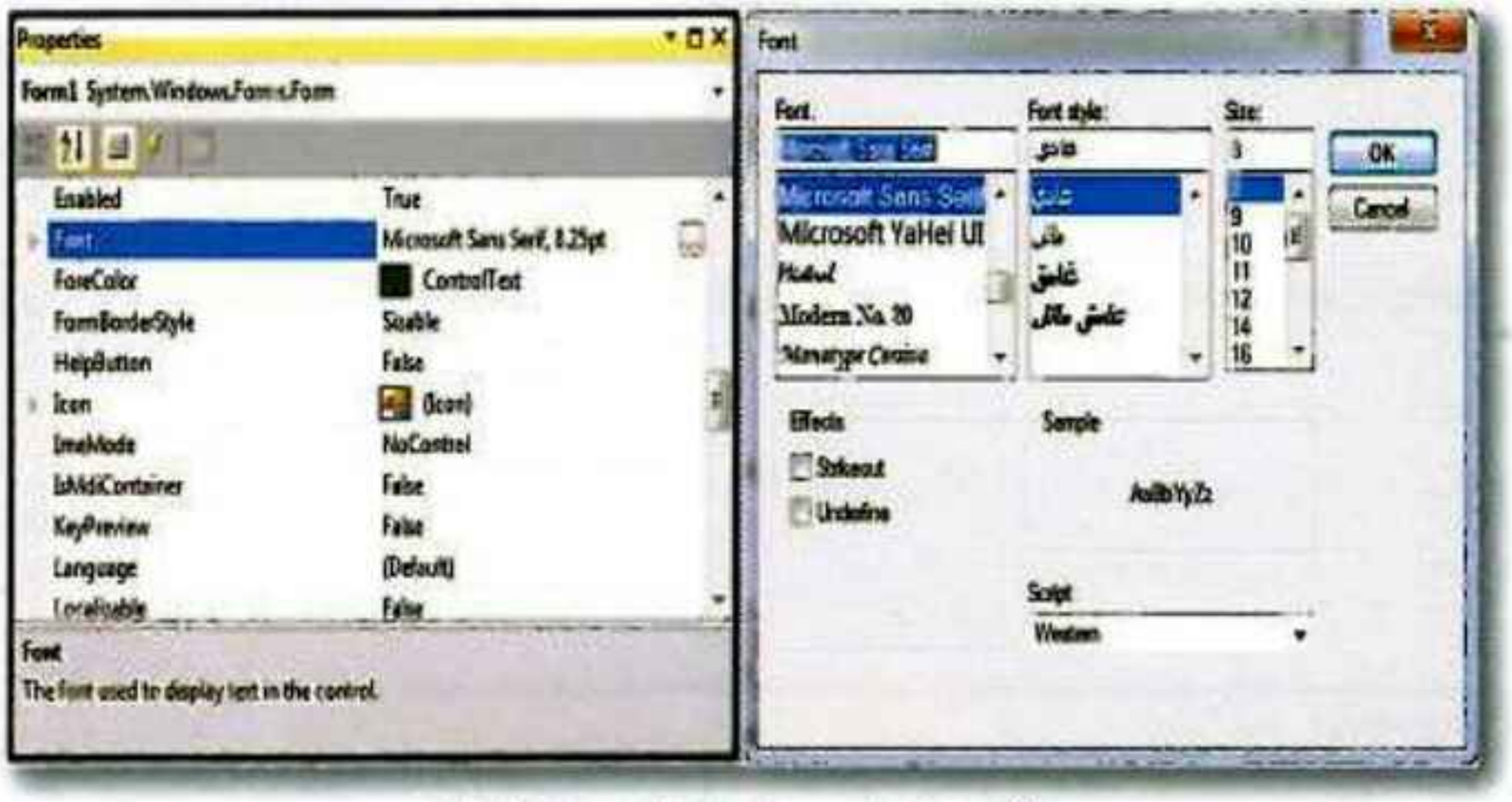
نجد في خانة إدخال قيمة الخاصية مجموعة من الاختيارات نقوم باختيار إحداها في ضبط خاصية المشاهدة كما في الشكل (٧-٥) (visible)، أي هل النموذج أو الأداة تظهر في الواجهة أم لا؟ حيث يعرض خياران إما (False) أو (True).



شكل (٧-٥) : خاصية المشاهدة

٣ - ظهور نافذة خيارات :

عند اختيار نوع الخط بالنقر على خاصية الخط (Font) كما في الشكل (٧-٦). تظهر نافذة خصائص نوع الخط.



شكل (٧-٦) : خاصية الخط ونافذة الخط

إثراء علمي

يقوم برنامج (فيجول بيسك ستوديو) بإعطاء قيم افتراضية للخواص، وذلك تسهيلاً للمبرمج. فمثلاً يعطي أسماء تلقائية لكل أداة تقوم برسمها، فعندما ترسم أداة تسمية لأول مرة فإن (فيجول بيسك ستوديو) يعطيها اسم (label1)، وعندما ترسم أداة التسمية مرة أخرى في النموذج نفسه فإن (فيجول بيسك ستوديو) يعطيها اسم (label2).. وهكذا لبقية الأدوات. وبإمكانك تغيير هذه الأسماء كما تريد.

ب ضبط خصائص الأداوت أثناء تشغيل البرنامج :

لتغيير خاصية الأدوات أثناء تنفيذ البرنامج فإننا نستخدم الصيغة التالية للوصول إلى تلك الخاصية :

```
القيمة الجديدة=الخاصية. اسم الأداة
```

مثلًا لتغيير خاصية (النص) في أداة مربع النص (Textbox1)نكتب مايلي :

```
Textbox1.Text= "مدرسة الرياض"
```

تذكر

يجب وضع نقطة بين اسم الأداة والخاصية التي نريد الوصول إليها.

ثالثا أدوات إدخال البيانات :

نستخدم أدوات إدخال البيانات للحصول على البيانات التي يجب على البرنامج معالجتها.

يقوم المستخدم بإدخال هذه البيانات بالكتابة أو الاختيار باستخدام هذه الأدوات، والشكل (٧-٧) يعرض بعضها.

شكل (٧-٧): مربع الأدوات

١ أداة مربع النص (TextBox): تتيح للمستخدم كتابة نص وتخزين النص في الخاصية (Text).

٢ أداة زر الخيار (RadioButton): تتيح للمستخدم انتقاء خيار واحد فقط من عدة خيارات، وتخزن قيمها في الخاصية (Checked).

٣ أداة مربع الأختيار (CheckBox): تتيح للمستخدم انتقاء عدة خيارات، وتخزن قيمها في الخاصية (Checkstate) والخاصية (Checked).

٤ أداة مربع القائمة (ListBox): تعرض قائمة مكونة من عناصر يختار المستخدم أحدها وتخزن خيار المستخدم في الخاصية (Text) أو (SelectedIndex).

٥ أداة الخانة المركبة (ComboBox): تعطي المستخدم حرية الاختيار من قائمة أو إدخال اختياره كتابة وتخزنها في الخاصية (Text).

وتختلف طريقة الحصول على البيانات من هذه الأدوات؛ لذلك سوف نتعرف على طريقة كل أداة على حدة:

١ مربع النص (Text Box):

إن البيانات التي نحصل عليها من مربع النص تختلف بحسب ما يدخله المستخدم، فقد تكون أرقامًا أو حروفًا.

للحصول على البيانات من مربع النص نستخدم الصيغة التالية:

```
Var1=TextBox.Text
```

حيث إن:

Var1 : متغير لتخزين البيانات فيه أياً كان نوعها عددية أم حرفية.

TextBox : اسم أداة مربع النص على النموذج.

Text : خاصية النص في أداة مربع النص التي تستقبل البيانات من المستخدم.

مثال :

للحصول على القيمة المدخلة لاسم المستخدم في مربع النص والمسمى (Textbox1) وتخزينها في المتغير (Username) نكتب الأمـر التالي:

```
Username = Textbox1.Text
```

في هذا المثال كما في الشكل (٧-٨) سوف يخزن في المتغير القيمة التالية:

```
Username="Administrator"
```

شكل (٧-٨) : أداة مربع النص

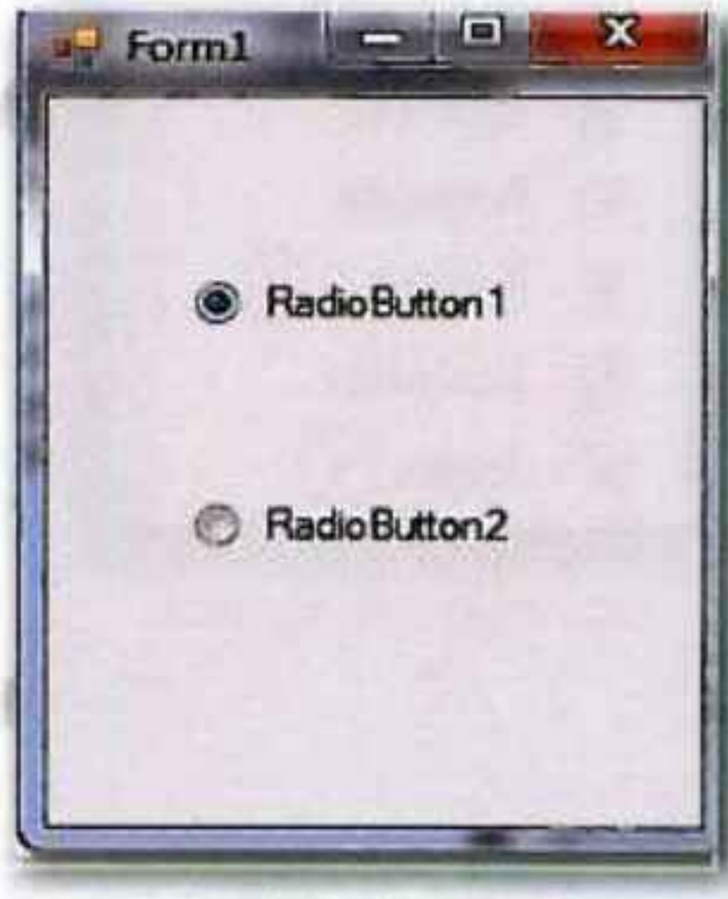
٢ أداة زر الخيار (RadioButton) :

إن البيانات التي نحصل عليها من زر الخيار كما في الشكل (٧-٩) هي بيانات منطقية تخزن في الخاصية (checked) وهي إحدى قيمتين، إما:

- أن الزر قد تم اختياره فقيمته عندئذ هي (True).
- أن الزر لم يتم اختياره فقيمته عندئذ هي (False).

والصيغة العامة للحصول على البيانات هي:

```
Var1=RaidoButton.Checked
```



شكل (٦-٩) : أداة زر الخيار

حيث إن :

Varl : متغير لتخزين البيانات من نوع منطقي.

RaidoButton : اسم أداة زر الخيار على النافذة.

Checked : خاصية أداة زر الخيار التي تستقبل البيانات من المستخدم، إما (True) أو (False).

مثال :

لمعرفة أن الزر المسمى (OB1) قد تم اختياره أم لا نكتب السطر التالي:

H = OB1.Checked

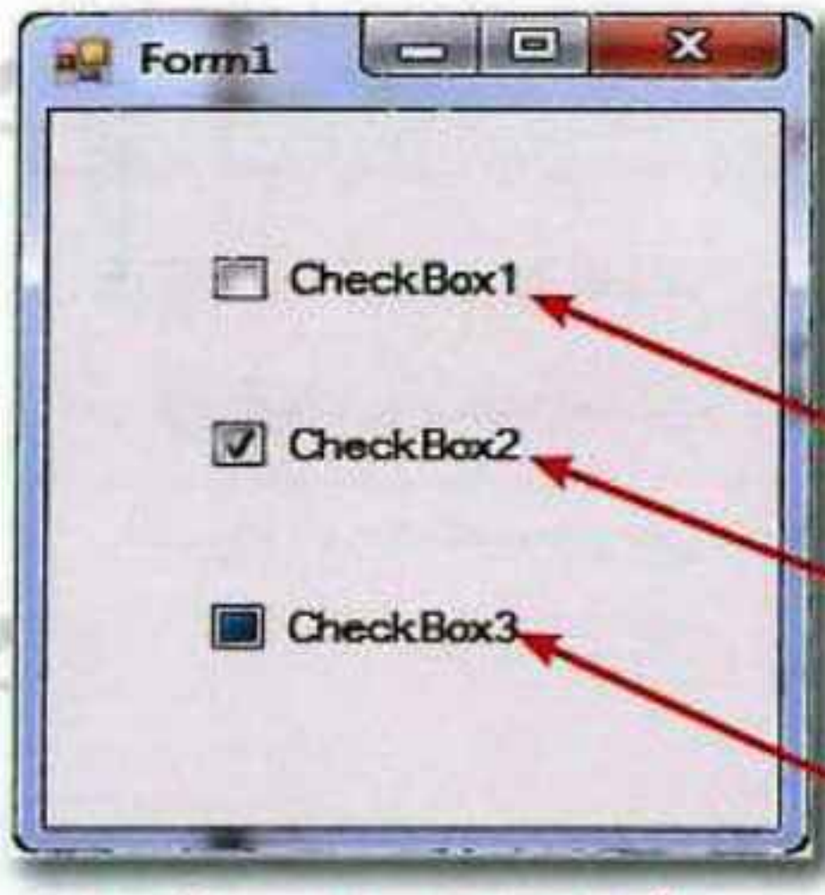
حيث إن (H) يمثل متغيراً منطقياً يُخزِّن إحدى القيمتين، إما صح (True) أوخطأ (False)

لمعرفة إن كان الزر قد تم اختياره أم لم يتم.

٣ أداة مربع الاختيار (CheckBox) :

إن البيانات التي نحصل عليها من مربع الاختيار كما في الشكل (٧-١٠) هي بيانات رقمية وليست منطقية كما في زر الخيار، وتخزن في الخاصية (checkstate)، وهي إحدى ثلاثة أشياء، إما:

- أن المستخدم لم يختر المربع فقيمته عندئذ هي (0).
- أن المستخدم اختيار المربع فقيمته عندئذ هي (1).
- أن المستخدم لا يستطيع اختيار المربع فقيمته عندئذ هي (2).



شكل (٧-١٠) : أداة مربع الاختيار

والصيغة العامة للحصول على البيانات هي :

```
Var1 = CheckBox.Checkstate
```

حيث إن :

Varl : متغير لتخزين البيانات فيه من نوع عددي.

CheckBox : اسم أداة مربع الاختيار على النافذة.

Checkstate : خاصية أداة مربع الاختيار التي تستقبل البيانات من المستخدم.

مثال:

لمعرفة أن مربع الاختيار المسمى (CheckBox2) قد تم اختياره أم لا نكتب السطر التالي:

```
A = CheckBox2 .Checkstate
```

حيث إن (A) يمثل متغيراً عددياً يخزّن أحد الأعداد التالية (2،1،0) لمعرفة إن كان مربع الاختيار قد تم اختياره، أم لم يتم، أم لا يمكن اختياره.

٤ أداة مربع القائمة (ListBox) :

البيانات الموجودة في أداة مربع القائمة مكونة من عدة عناصر، وللحصول على البيانات من القائمة عندما يختار المستخدم أحد العناصر يعني أحد شيئين:

- رقم العنصر في القائمة بواسطة الخاصية (SelectedIndex) .
- قيمته بواسطة الخاصية (Text) .

الصيغة العامة للحصول على رقم العنصر (SelectedIndex) هي:

```
Varl = ListBox.SelectedIndex
```

حيث إن :

Varl : متغير لتخزين البيانات فيه من نوع رقمي.

ListBox : اسم أداة مربع القائمة على النافذة.

SelectedIndex : خاصية أداة مربع القائمة التي تحدد رقم العنصر الذي اختاره المستخدم.

الصيغة العامة للحصول على قيمة العنصر (Text) هي:

```
Var1=ListBox.Text
```

حيث إن :

Varl : متغير لتخزين البيانات فيه أياً كان نوعها عددية أم حرفية.

ListBox : اسم أداة مربع القائمة على النافذة.

Text : خاصية أداة مربع القائمة التي تحدد قيمة العنصر الذي اختاره المستخدم.

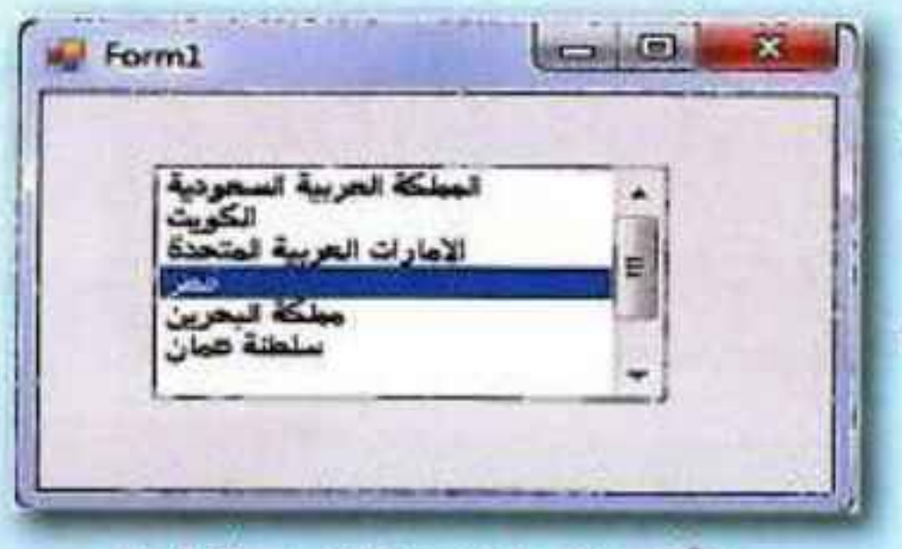
مثال ١ :

لمعرفة رقم العنصر الذي اختاره المستخدم من القائمة (ListBox1) التي تمثل هنا أسماء الدول:

```
C = ListBox1.SelectedIndex
```

حيث إن (C) يمثل متغيراً عددياً يخزّن رقم العنصر الذي اختاره المستخدم.

وفي هذا المثال سوف يكون مخزن في المتغير (C) الرقم (3). كما في الشكل (٧-١١).



شكل (٧-١١) : أداة مربع القائمة

مثال ٢ :

لمعرفة قيمة العنصر الذي اختاره المستخدم في القائمة (ListBox1):

```
D = ListBox1.Text
```

حيث إن (D) يمثل متغيراً يخزّن قيمة العنصر في القائمة.

وفي هذا المثال تكون القيمة: "الامارات العربية المتحدة" = D ،

كما في الشكل (٧-١٢).

شكل (٧-١٢) : أداة مربع القائمة

٥ أداة الخانة المركبة (ComoBox) :

أداة الخانة المركبة تجمع بين ميزات أداة النص وأداة مربع القائمة، حيث يستطيع المستخدم أن يختار من القائمة أو يكتب قيمة جديدة .

للحصول على البيانات من أداة الخانة المركبة نستخدم الصيغة التالية:

```
Varl = ComboBox.Text
```

حيث إن :

Varl : متغير لتخزين البيانات فيه أياً كان نوعها عددية أم حرفية.

ComboBox: اسم أداة الخانة المركبة على النافذة.

Text : خاصية أداة الخانة المركبة التي تستقبل البيانات من المستخدم سواء باختياره من القائمة أو بإدخاله للبيانات بالكتابة مباشرة.

مثال :

للحصول على القيمة التي أدخلها المستخدم أو اختارها من القائمة في أداة الخانة المركبة (Combobox1) وتخزينها في المتغير (C) نكتب الأمر التالي:

```
C = Combobox1.Text
```

في هذا المثال سوف يخزن في المتغير (C) الدولة التي يختارها المستخدم، كما في الشكل (٧-١٢).

شكل (٧-١٢) : أداة الخانة المركبة

رابعًا أدوات إخراج المعلومات :

نستخدم أدوات إخراج المعلومات لإظهار المعلومات للمستخدم على الواجهة بعد أن عالج البرنامج البيانات التي أدخلها المستخدم، ومن هذه الأدوات :

١. أداة مربع النص (TextBox) : وتخرج المعلومات بواسطة الخاصية (Text).
٢. أداة التسمية (Label) : وتخرج المعلومات بواسطة الخاصية (Text) .

١ طريقة إخراج المعلومات إلى مربع النص (TextBox) :

لإخراج المعلومات إلى مربع النص نستخدم الصيغة التالية :

```
TextBox . Text = Varl
```

حيث إن :

TextBox : اسم أداة مربع النص على النافذة.

Text : خاصية النص في أداة مربع النص التي سوف نخزن فيها قيمة المتغير (Var1) .

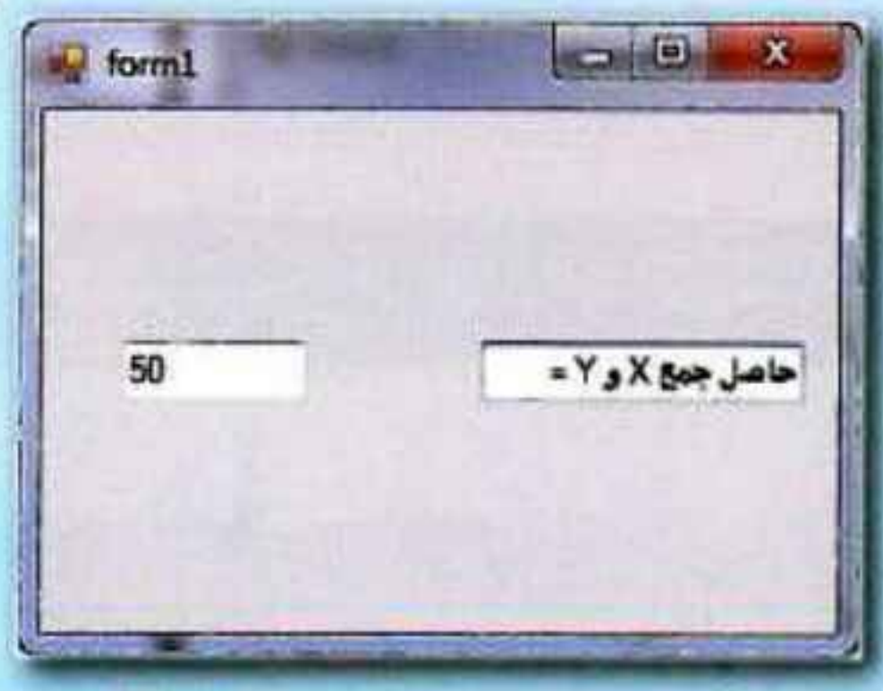
Varl : اسم المتغير .

مثال :

لإظهار حاصل جمع عددين (X+Y) في الأداة المسماة (Text4) نكتب السطر التالي :

```
Text4.Text = X+Y
```

في هذا المثال كما في الشكل (٧-١٤) يظهر الناتج.



شكل (٧-١٤) : استخدام أداة (Text)

٢ طريقة إخراج المعلومات إلى أداة التسمية (Label) :

لإخراج المعلومات إلى أداة التسمية نستخدم الصيغة التالية :

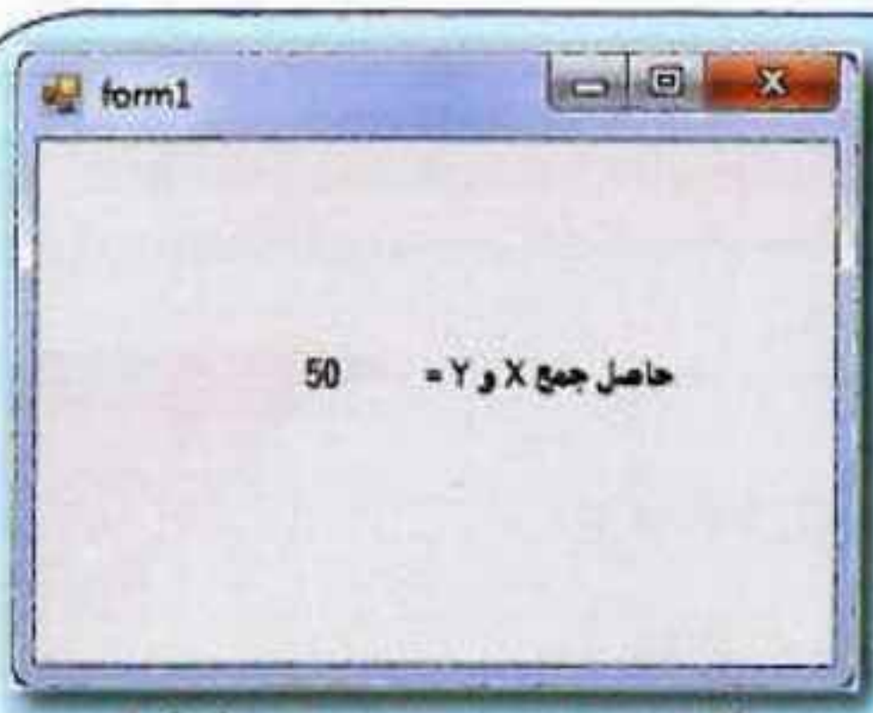
```
Label. Text = Var1
```

مثال :

لإظهار حاصل جمع عددين (X+Y) في الأداة المسماة (Lable1) نكتب السطر التالي :

```
Label1.Text = X+Y
```

في هذا المثال كما في الشكل (٧-١٥) يظهر الناتج.



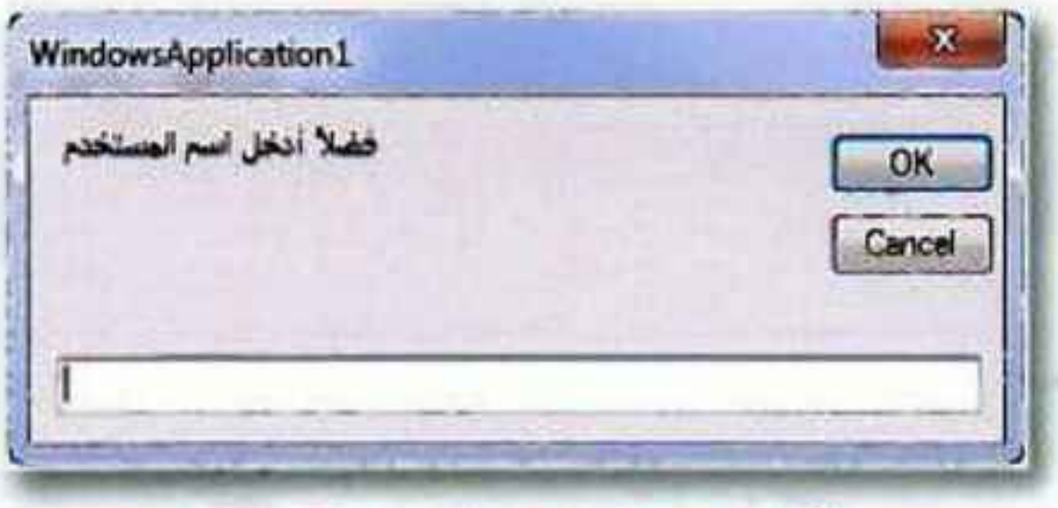
شكل (٧-١٥) : استخدام أداة (Label)

٧-٦ بعض الأوامر الأساسية للغة (فيجول بيسك ستوديو)

هناك أوامر داخلية في (فيجول بيسك ستوديو) تظهر للمستخدم نافذة مصممة سابقاً من قبل الشركة المنتجة للغة البرمجة، سواء لإدخال البيانات أو إخراج المعلومات، ومن هذه الأوامر:

أولاً أوامر إدخال البيانات وإخراج المعلومات :

١ إدخال البيانات بواسطة الأمر (InputBox) :



شكل (٧-١٦) : نافذة أمر (InputBox)

يُظهر هذا الأمر نافذة صغيرة غير النافذة الرئيسة في البرنامج تحتوي على مربع نص وزر أمر كما في الشكل (٧-١٦) ليدخل المستخدم البيانات التي يريدها في مربع النص ثم يضغط على زر الأمر. لإنشاء هذه النافذة يجب أن نكتب الأمر الخاص بها، وصيغته كالتالي:

```
Var1 = InputBox (message)
```

حيث إن :

Var1 : اسم المتغير الذي سوف تخزن فيه البيانات التي أدخلها المستخدم وقد تكون حرفية أو رقمية.

InputBox: أمر إنشاء هذه النافذة.

Message: النص الثابت الذي يظهر في النافذة ويوضع بين أقواس اقتباس هكذا " ".

مثال :

لو أردنا أن نطلب من المستخدم إدخال اسم المستخدم فإننا نكتب:

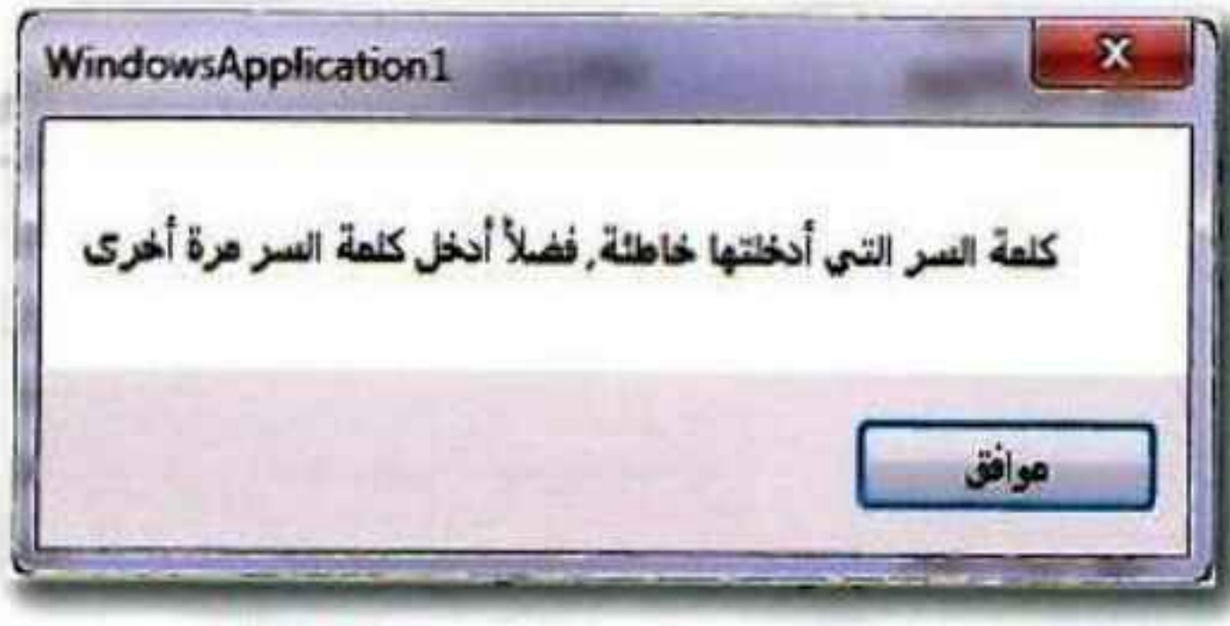
```
Username=InputBox("فضلاً أدخل اسم المستخدم")
```

٢ إخراج المعلومات بواسطة الأمر (MsgBox) :

يظهر هذا الأمر نافذة صغيرة غير النافذة الرئيسة في البرنامج تحتوي على المعلومات التي نريد للمستخدم قراءتها كما في الشكل (٧-١٧)، ولإنشاء هذه النافذة يجب أن نكتب الأمر الخاص بها:

وصيغته كالتالي:

```
MsgBox ( message )
```



شكل (٧-١٧) : نافذة أمر (MsgBox)

حيث إن :

MsgBox: أمر إنشاء هذه النافذة.

message: اسم المتغير أو نص ثابت يوضع بين أقواس اقتباس هكذا " " ليظهر في النافذة.

مثال :

لو أردنا أن نخبر المستخدم أن كلمة السر التي أدخلها خاطئة فإننا نكتب:

```
MsgBox ("كلمة السر التي أدخلتها خاطئة، فضلاً أدخل كلمة السر مرة أخرى")
```

وإذا أردنا عرض قيمة المتغير A فنكتب :

```
MsgBox (A)
```

ثانيا أمر الإسناد :

يقصد به تخزين قيمة معينة داخل متغير، وقد تكون هذه القيمة عدد أو عملية حسابية أو سلسلة حرفية.

وصيغته كالتالي:

```
Var1= Value
```

تفكر

عند إسناد سلسلة نصية إلى متغير حرفي نضعها داخل علامتي اقتباس .

حيث إن :

Var1 : اسم المتغير.

Value : القيمة التي نريد تخزينها في المتغير.

مثال :

لو أردنا تخزين اسم في متغير حرفي تم تعريفه مسبقاً : Name = "Nor"

لو أردنا تخزين رقم في متغير عددي تم تعريفه مسبقاً: Num=10

لو أردنا تخزين ناتج عملية حسابية في متغير عددي تم تعريفه مسبقاً: X= V+20×I

ثالثاً الجمل الشرطية :

يوجد في لغة (فيجول بيسك ستوديو) عدة جمل للتحقق من الشرط وهي :

١ الجملة الشرطية (IF) :

وتعد من أهم الأوامر في البرمجة، وتوجد في أغلب لغات البرمجة. ونستخدمها في البرنامج عند حاجتنا إلى اتخاذ قرارات مختلفة على حسب شرط معين، مثلاً إن كانت درجة الطالب أكبر من أو تساوي (٦٠) فهو ناجح، وإن كانت أقل من (٦٠) فهو غير مجتاز.

وتحتوي لغة (فيجول بيسك ستوديو) على عدة صيغ، منها:

أ صيغة (IF-THEN) :

IF condition THEN statement

فإذا تحقق الشرط (condition)، أي كان صحيحاً (True) فسيتم تنفيذ الأمر (statement)، وإذا لم يكن الشرط صحيحاً فلن يتم تنفيذ أي أمر.

مثال :

IF grade >= 60 THEN Result = "ناجح"

ب صيغة (IF-THEN-END IF):

```
IF condition THEN
...
statements
...
END IF
```

هذه الصيغة هي الصيغة الأولى نفسها، إلا أنه سيتم تنفيذ أكثر من أمر إذا كان الشرط صحيحاً؛ لذلك احتجنا إلى وضع كلمة (END IF) لتوضيح نهاية مجموعة الأوامر التي سيتم تنفيذها.

مثال:

```
IF grade > = 60 THEN
Result = "ناجح"
Text1 = "مبروك"
END IF
```

جـ صيغة (IF-THEN-ELSE):

```
IF condition THEN
...
statements1
...
ELSE
...
statements2
END IF
```

تختلف هذه الصيغة عن الصيغ السابقة، حيث يتم تنفيذ أوامر (statements1) في حالة كون الشرط صحيحًا (True)، وفي حالة كون الشرط غير صحيح (False) يتم تنفيذ مجموعة الأوامر (statements2).

مثال:

```
IF grade > = 60 THEN
Result = "ناجح"
ELSE
Result = "غير مجتاز"
END IF
```

د صيغة (IF-THEN-ELSEIF):

```
IF condition1 THEN
...
statements1
...
ELSEIF condition2  THEN
...
statements2
...
ELSE
...
Statements3
...
END IF
```

تختلف هذه الصيغة عن الصيغة السابقة في وجود أكثر من شرط يتم التحقق منه.
إذا كان الشرط الأول (condition1) صحيحاً فإنه ينفذ الأوامر (statments1) فقط.
أما إذا كان الشرط الأول غير صحيح فإنه يختبر شرطًا جديدًا وهو (condition2) وإذا كان صحيح فإنه ينفذ مجموعة الأوامر (statements2) فقط.
أما إذا كان الشرط الثاني غير صحيح فإنه ينفذ مجموعة الأوامر (statements3). وقد يكون لدينا شرط ثالث ورابع وهكذا.

مثال :

```
IF grade > = 90 THEN
Result="ممتاز"
ELSEIF grade > = 80 THEN
Result="جيد جدا"
ELSEIF grade > =70 THEN
Result="جيد"
ELSEIF grade > = 60 THEN
Result="مقبول"
ELSE
Result="غير مجتاز"
END IF
```

٢ الجملة الشرطية (Select Case) :

تستخدم هذه الجملة إذا كان هناك عدة احتمالات للشرط. فبدلاً من استخدام جملة (IF) طويلة ومعقدة تقوم هذه الجملة بالعمل نفسه ولكن بطريقة أسهل. حيث تختبر هذه الجملة تعبيرًا أو شرطًا معينًا قد يكون لقيمته أكثر من احتمال.

وصيغتها :

```
SELECT CASE expression
CASE prob1
...
statements1
...
CASE prob2
...
statements2
...
[CASE ELSE
...
statements3
...]
END SELECT
```

حيث إن :

SELECT CASE : بداية الجملة.

expression : الشرط أو التعبير الذي نريد اختبار قيمته، وقد يكون متغيرًا أو عملية حسابية أو عملية منطقية.

CASE : توضع قبل كل احتمال.

prob1،prob2،....: القيم المحتملة للتعبير.

statements1 : الأوامر التي تنفذ في حالة تحقق القيمة.

CASE ELSE : إذا لم يتحقق أي احتمال من الاحتمالات السابقة فسوف تنفذ الأوامر التي بعد هذه العبارة، وهي اختيارية، أي إذا لم نكن بحاجة لها فلا يجب استخدامها.

END SELECT : نهاية الجملة.

تحكم

وجود الجزء (CASE ELSE) داخل الأقواس [ ] وذلك لأنه جزء اختياري من الصيغة إذا كنا بحاجة إليه نضعه، وعدم وجوده لا يؤثر على صحة الجملة.

طريقة عمل هذه الجملة كالتالي:

يقوم البرنامج بتقييم التعبير (expression)، ثم يقارنه مع الاحتمالات الواردة عند كل كلمة (CASE prob1،prob2،....)، فإذا وافق قيمة التعبير أحد هذه الاحتمالات فسوف ينفذ الأوامر التي جاءت بعد الاحتمال الصحيح وحتى جملة (CASE) التالية.

فلو كان التعبير يوافق الاحتمال الأول (prob1) فإن البرنامج سوف ينفذ مجموعة الأوامر (statements1) فقط، ويذهب إلى نهاية الجملة.

أما إذا لم يوافق التعبير أياً من الاحتمالات الموجودة، فإذا كان لدينا (CASE ELSE) فإن البرنامج سوف ينفذ مجموعة الأوامر التي تأتي بعده، وإذا لم يكن لدينا (CASE ELSE) (لأنه اختياري لا يلزم وجوده دائماً) فإن الجملة تنتهي دون تنفيذ أي أوامر.

تحكم

يجب التأكد من أن نوع بيانات (expression) هو نفسه نوع البيانات الموجودة في الاحتمالات.

مثال :

لو أردنا تطبيق المثال السابق نفسه في جملة (IF-THEN-ELSEIF) ولكن باستخدام جملة SELECT CASE

```
SELECT CASE grade
CASE 90 to  100
Result="ممتاز"
CASE 80 to 89
Result="جيد جدا"
CASE 70 to 79
Result="جيد"
CASE 60 to 69
Result="مقبول"
CASE ELSE
Result = "غير مجتاز"
END SELECT
```

رابعا حلقات التكرار :

هو من أهم أوامر البرمجة التي تساعدنا على تكرار مجموعة من الأوامر الأخرى عدة مرات. ويوجد في لغة (فيجول بيسك ستوديو) عدة أوامر للتكرار ومن أهمها:

١ الأمر (For .. Next) :

يكرّر هذا الأمر مجموعة من الأوامر بعدد من المرات محدد ومعروف مسبقاً.

صيغته:

```
FOR counter=start TO end [STEP step]
.........
statements
.......
NEXT
```

حيث إن :

Counter : هو متغير يخزن فيه عدد مرات التكرار يبدأ من قيمة أولية ويتغير إلى أن يصل إلى القيمة النهائية المحددة له، ويسمى هذا المتغير بالعدّاد.

Start : القيمة الأولية التي يبدأ بها العدّاد.

end : القيمة النهائية التي يجب أن يتوقف عندها العدّاد.

Step : القيمة التي يتم بها زيادة العدّاد في كل دورة تكرار. وهي اختيارية، فإذا لم نذكرها فإن الزيادة سوف تكون (1).

Statements : مجموعة الأوامر أو قد يكون أمراً واحداً تُنفذ بعدد مرات التكرار.

NEXT : نهاية جملة التكرار، أي أن الأوامر التي تأتي بعده لا تدخل في التكرار.

مثال :

لو أردنا جمع الأعداد من (1) إلى (10) وتخزينها داخل المتغير (sum):

يمكن أن نكتب أوامر بهذه الطريقة:

يجب أن نضع قيمة ابتدائية في المتغير قبل أن نجمع عليه

```
Sum=0
```

ثم نبدأ بجمع الأعداد واحداً تلو الآخر، وهذا يتطلب منا أن نكتب (10) أوامر كالتالي:

```
sum=sum+1
sum=sum+2
.....
sum=sum+10
```

الأفضل من هذه الطريقة أن نستخدم جملة تكرار كالتالي:

```
For count=1 to 10
sum=sum+count
Next
```

حيث إننا لم نحدد قيمة (Step) هنا فإن الزيادة سوف تكون (1) في كل مرة. أي يبدأ العدّاد من القيمة (1) ويزداد إلى أن يصل إلى القيمة (10). وفي كل مرة يجمع هذه القيم على المتغير (sum). وبعد تنفيذ التكرار سوف يكون لدينا في المتغير (sum) مجموع الأعداد من (1) إلى (10).

مثال :

لو أردنا جمع الأعداد الفردية من (1) إلى (11) فإننا سوف نستخدم الطريقة السابقة نفسها، وسنقوم بتحديد مقدار الزيادة على أن يكون (2) كالتالي:

```
sum=0
For count=1 to 11 STEP 2
sum=sum+count
Next
```

٢ الأمر (DO WHILE) :

نستخدم هذا الأمر إذا كان عدد مرات التكرار غير محدد، ولكنّ لدينا شرطًا هو الـذي يحدد متى ينتهي التكـرار، أي أنـه متى مـا كان الشرط صحيحاً نفذنا الأوامر واستمر التكـرار، ومتى ما صـار الشـرط غير صحيح توقف التكرار.

إثارة التفكير

ما الذي سيجعل الشرط غير صحيح؟ لابد أننا سوف نقوم بعمليات داخل التكرار تؤثر على الشرط.

صيغته:

```
Do While condition
.....
statements
.....
Loop
```

حيث إن :

condition : الشرط الذي يتم التحقق منه، ثم تنفيذ التكرار إذا كان صحيحاً والتوقف إذا كان خاطئاً.

statements: مجموعة الأوامر التي تنفذ داخل التكرار.

مثال ١:

```
A=1
sum=0
Do While A < = 10
sum=sum+A
A=A+1
Loop
```

في هذا المثال تُجمع الأرقام من (1) إلى (10) كما في المثال السابق. ونلاحظ هنا أن شرط التوقف هو وصول قيمة المتغير (A) إلى (10).

مثال ٢:

لو أردنا جمع الأعداد الزوجية من (0) إلى (10).

```
A=0
sum=0
Do While A < =10
sum=sum+A
A=A+2
Loop
```

خامسًا المصفوفات:

لو كان لديك درجات (100) طالب تريد عمل بعض الإحصاءات عليها كمعرفة المتوسط وأعلى درجة وأقل درجة. فأين سوف تخزن هذه الدرجات؟ هل سوف تعرّف (100) متغير لتخزينها؟ يبدو هذا غير منطقي، أليس كذلك؟

يوجد في لغة (فيجول بيسك ستوديو) وفي أغلب لغات البرمجة ما يسهل علينا عملية تعريف عدد كبير من المتغيرات تشترك في كونها تمثل نوع البيانات نفسه وهي المصفوفات.

المصفوفة (Array) هي مجموعة من المتغيرات لها الاسم نفسه ونوع البيانات نفسه ويتم تعريفها في جملة واحدة.

صيغتها:

```
Dim var1( n ) As Type
```

حيث إن :

var1 : اسم المصفوفة.

n : (عدد عناصر المصفوفة - 1).

Type : نوع البيانات المخزنة في العناصر.

مثال:

```
Dim Grades( 99 ) As Integer
```

هنا عرَّفنا مصفوفة لتخزين درجات (100) طالب.

أليس هذا أفضل من تعريف (100) متغير؟

١ فوائد المصفوفات :

كما لاحظت في المثال السابق، فإن استخدام المصفوفة قد وفر علينا كثيرًا من الوقت والجهد الذي كنا سنبذله في تعريف (100) متغير ومعالجة كل متغير على حدة. فالمصفوفات سهلت لنا هذه المهمة، ونستطيع باستخدام أوامر التكرار أن نتعامل مع المصفوفات بسهولة. كما يؤدي استخدام المصفوفات إلى صغر حجم البرنامج.

٢ التعامل مع المصفوفات :

للوصول إلى عنصر من عناصر المصفوفة نكتب اسم المصفوفة وبين قوسين رقم العنصر، ولكن يجب التنبه إلى أن ترقيم العناصر في المصفوفة يبدأ من الصفر أي أن أول عنصر في المصفوفة رقمه (0) ثم العنصر الثاني (1) وهكذا إلى آخر عنصر في المصفوفة الذي يكون رقمه عدد عناصر المصفوفة (-1).

مثال: لو عرفنا مصفوفة فيها (10) أعداد كالتالي: `Dim A(9) AS Integer`

وخزنا فيها مجموعة من الأرقام، سيكون شكل المصفوفة كالتالي:

المصفوفة A									
رقم العنصر	0	1	2	3	4	5	6	7	8	9
القيمة	4	3	5	6	2	15	7	9	12	8

لو أردنا تغيير قيمة العنصر الخامس لكتبنا:

$$A(4)=10$$

تذكر

أن تبدأ العـدّاد مـن الصفـر وتنهيـه بعدد العناصر - 1 عند استخدامك للمصفوفات .

غالباً ما تُعالج جميع عناصر المصفوفة بالتسلسل، أي واحداً تلو الآخر، وما يسهل علينا هذه المعالجة هو استخدام حلقات التكرار، حيث نجعل العدّاد يمثل رقم العنصر كما في المثال التالي:

مثال:

لقراءة درجات (100) طالب نقوم بالتالي:

```
Dim Grades(99) As Integer
FOR count=0 To 99
Grades(count)=InputBox ("أدخل الدرجة")
NEXT
```

لو أردنا أن نجد متوسط درجات الطلاب من المثال السابق، فيجب علينا أولاً أن نجمع جميع الدرجات ثم نقسم على عدد الطلاب.

نعرف أولاً متغيرًا لحساب المجموع وآخر لحساب المعدل:

```
Dim sum As Integer , average As Single
sum=0
For count=0 To 99
sum=sum+Grades (count )
Next
average=sum/100
```

لو أردنا أن نجد أعلى درجة من درجات الطلاب .

نعرف أولاً متغيرًا لتخزين أعلى درجة:

```
Dim max As Integer
max=0
For count=0 To 99
IF Grades(count)>max     THEN max=Grades(count)
Next
```

مشروع الوحدة

المشروع الأول :

قم بتصميم برنامج لإيجاد القاسم المشترك الأكبر لعددين باستخدام نظرية اقليدس.

المشروع الثاني :

قم بتصميم برنامج لمغسلة ملابس تقوم فيه بإدخال اسم العميل ثم اختيار نوع الملابس وأسعارها ثم عرض اسم المستخدم وقائمة ملابسه مع أسعارها وإجمالي فاتورته انظر الشكل للنموذج المطلوب عرضه :

اسم العميل : محمد أحمد عبدالله

الملابس :

القطعة	العدد	السعر الفردي	السعر الإجمالي
١.ثوب	٥	٣	١٥
٢. غترة	٢	٢	٤

إجمالي القطع : ١٠ إجمالي السعر : ١٤

باستخدام برنامج فيجول بيسك ستوديو قم باختيار أحد المشروعات أعلاه، وكتابة تقرير عن المشروع يشمل :

١. مقدمة عن التطبيق (الفكرة - الهدف).
٢. خطوات حل المسألة.
٣. خوارزم البرنامج.
٤. صور الواجهات المصممة وعمل مكونات كل واجهة.
٥. النص البرمجي للبرنامج.

سوف نقوم باختيار المشروع الأول.

١. مقدمة عن التطبيق:

هذا التطبيق لحساب القاسم المشترك الأكبر لعددين طبيعيين باستخدام نظرية إقليدس.

٢. مدخلات البرنامج: العددين x,y.

٣. مخرجات البرنامج: القاسم المشترك الأكبر gcd.

٤. عمليات المعالجة: إيجاد القاسم المشترك الأكبر للعددين x,y وذلك باستخدام قانون إقليدس.

٥. الخوارزمية:

- ادخل العددين x,y
- اجعل i=1
- إذا كانت i=<x اجعل i=i+1
- اجعل b=x/i
- اجعل c=y/i
- إذا كانت b=c=0 اجعل gcd=i وإلا اذهب للخطوة 3
- اطبع gcd
- النهاية

٦. واجهة البرنامج:

7. النص البرمجي:

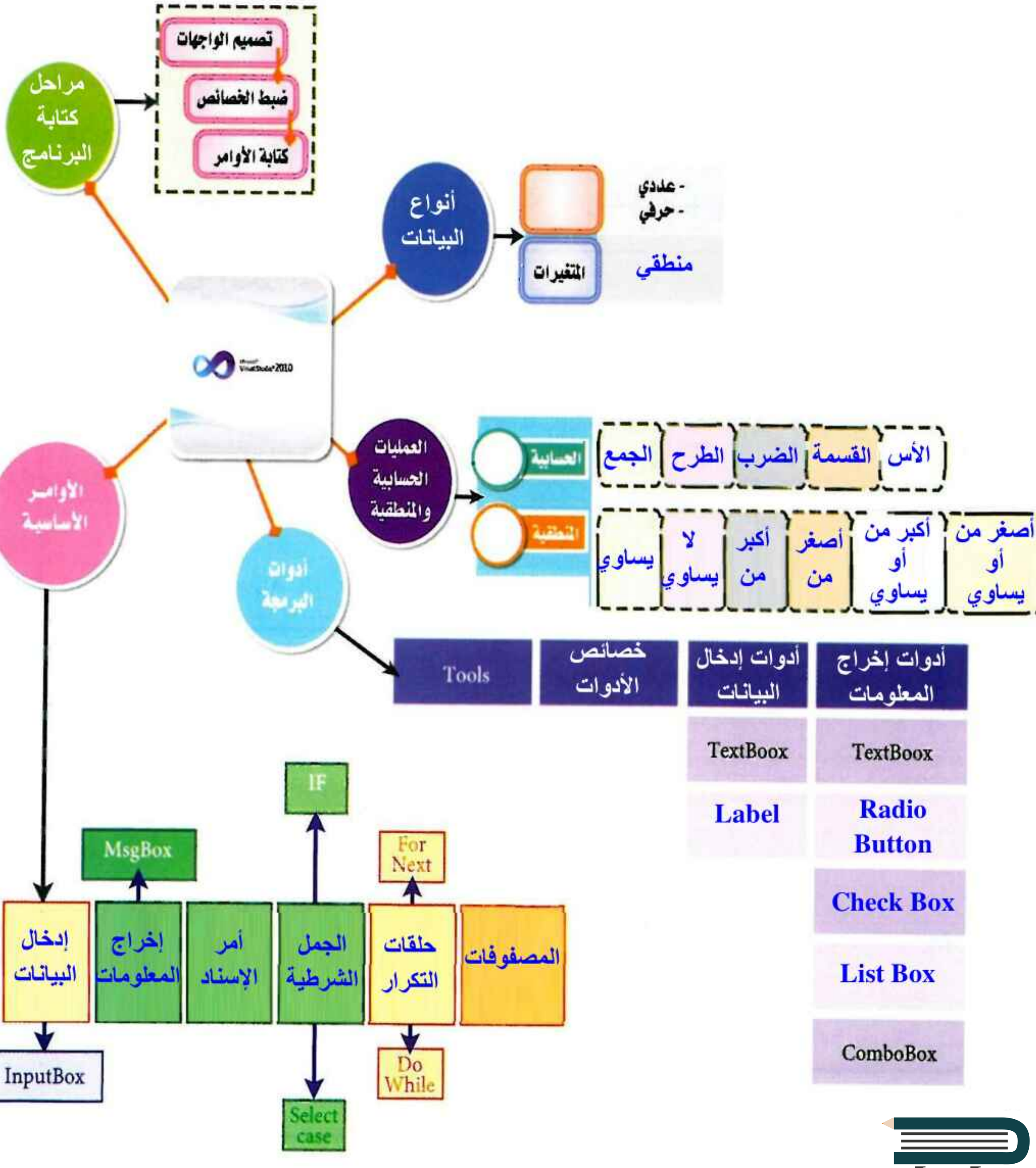
```
Public Class Form1

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e A
        Dim x, y, i, gcd As Integer
        x = TextBox1.Text
        y = TextBox2.Text
        i = 1
        Do While i <= x
            i = i + 1
            If (x Mod i) = 0 And (y Mod i) = 0 Then
                gcd = i
                Label4.Text = gcd
            End If
        Loop
    End Sub
End Class
```

8. نتيجة تشغيل البرنامج:

خارطة مفاهيم الوحدة

أكمل خارطة المفاهيم أدناه باستخدام العبارات والمصطلحات التي تعلمتها في الوحدة :



دليل الدراسة

مفردات الوحدة	المفاهيم الرئيسة
■ مراحل كتابة البرنامج بلغة (فيجول بيسك ستديو).	■ تصميم الواجهات. ■ ضبط الخصائص. ■ كتابة الأوامر البرمجية.
■ تعامل (فيجول بيسك ستديو) مع البيانات.	■ الثوابت وأنواعها وتعريفها. ■ المتغيرات أنواعها وتعريفها. ■ شروط تسمية المتغيرات. ■ أنواع البيانات.
■ العمليات الحسابية والمنطقية.	■ العمليات الحسابية: الجمع – الطرح – الضرب – القسمة – التربيع. ■ العمليات المنطقية: يساوي – لا يساوي – أكبر من – أقل من – أكبر من أو يساوي – أصغر من أو يساوي.
■ أدوات البرمجة بلغة (فيجول بيسك ستديو).	■ الأدوات «Tools». ■ الخصائص «Properties». ■ أدوات إدخال البيانات. ■ أدوات إخراج المعلومات.
■ الأوامر الأساسية.	■ إدخال البيانات. ■ إخراج المعلومات. ■ أوامر الإسناد. ■ الجمل الشرطية. ■ حلقات التكرار.

تمرينات

س١ ما مراحل كتابة البرنامج بلغة (فيجول بيسك ستوديو)؟

- تصميم الواجهات.
- ضبط خصائص الواجهات.
- كتابة أوامر البرمجة.

س٢ ماذا نسمي أماكن تخزين البيانات في الذاكرة الرئيسة؟

هياكل البيانات.

س٣ ماذا يعني الأمر التالي: Dim Number As Integer؟

هذا من أوامر برنامج بلغة فيجوال بيسك أي تعريف اللمصفوفات والثوابت الصحيحة أي تستخدم الأمر Dim في طريقة تعريف الثوابت والذي تستخدمه أيضاً لتعريف المصفوفة.

س٤ ما الفرق بين الثوابت والمتغيرات؟

الثابت: هو إعطاء اسم لقيمة معينة ويستخدم داخل البرنامج، ولا يمكن تغيير هذه القيمة عند تنفيذ البرنامج.

المتغير: هو مكان في الذاكرة الرئيسية تخزن فيه بيانات وتعطى اسماً معيناً.

تـمـريـنـات

س٥ هل الأسماء التالية يمكن استخدامها لتسمية المتغيرات :

2ABC, 123, AB2, AB_2, Num one, While, aBxY, Case

لا؛ ليست كلها Case, While لا يمكن استخدامها؛ لأنها محجوزة للغة البرمجة و Num one يحتوي على مسافة و 2ABC يبدأ برقم، أما البقية فيمكن استخدامهم.

س٦ بافتراض المتغيرات والقيم التالية : X=20, Y=33, Z=9, A=2

ما نواتج العمليات الحسابية التالية :

X+Z*A^2

عملية الأس أولاً: 4 *9 +20

عملية الجمع ثانياً: 4 * 29

أخيراً: عملية الضرب: 116

(Y+X/A+1) / (Z+A)

العمليات التي داخل الأقواس أولاً:

(33+20 /2+1) /(9+2)

11/ (53/11)

11/ 17.7

ثانياً: عملية القسمة: 1.6

.....

تمرينات

X*5^A

عملية الأس أولاً: 25 *20

ثم عملية الضرب: 500

س٧ حوِّل العمليات الجبرية التالية إلى صيغة برمجية:

$\frac{x+y}{9*3} + M^x$

3 * 9 /(X + Y) M ^ X

$z\,x+4+y$

X^ 2 + Y + 4

$3y^{x+6}$

X^ y (x + 6)

.....

تمرينات

س٨ ماذا تسمى أجزاء البرامج الجاهزة التي توفرها لغة (فيجول بيسك ستوديو) لتوفر على المبرمج الجهد والوقت؟

الأدوات: Tools.

س٩ كيف نغير النص المكتوب على زر أمر اسمه (Button)؟

سيظهر لنا يمين الشاشة Caption أمامها Command 1 نحذفها ونكتب النص الذي نريد.

س١٠ ماذا نسمي الأدوات التي تستقبل البيانات من المستخدم؟ اذكر ثلاثاً منها، واذكر متى تستخدم.

أداة مربع النص: تتيح للمستخدم كتابة نص واستخدام وتخزين النص في الخاصية Text.

أداة زر الخيار: تتيح للمستخدم انتقاء خيار واحد فقط من عدة خيارات، وتخزن قيمتها في الخاصية Checked.

أداة الخانة المركبة: تعطي المستخدم حرية الاختيار من قائمة أو إدخال اختياره كتابة وتخزينها في الخاصية Text.

س١١ ضع علامة (✓) أمام العبارة الصحيحة ، وعلامة (✗) أمام العبارة الخاطئة، مع تصحيح الخطأ :

أ) يجب علينا عند البدء في عمل برنامج بلغة (فيجول بيسك ستوديو) كتابة أوامر البرمجة أولاً. (✗)

يجب علينا تصميم الواجهات أولاً.

ب) يمكن للبرنامج أن يغير قيمة الثابت عند تنفيذ عملية حسابية. (✗)

لا يمكن تغير قيمة الثابت في البرنامج.

جـ) نتائج العمليات المنطقية هي دائماً أرقام. (✗)

ناتج العمليات المنطقية دائماً (true) أو (false).

د) ننفِّذ عمليات الضرب والقسمة قبل عمليات الجمع والطرح . (✓)

هـ) إذا أردنا المستخدم أن يدخل رقم هاتفه نستخدم أداة مربع الاختيار. (✗)

أداة مربع النص (text box).

س١٢ اذكر ثلاث طرق لإخراج معلومات للمستخدم.

١. طريقة إخراج المعلومات إلى مربع النص Text Box.

٢. طريقة إخراج المعلومات إلى أداة التسمية Label.

٣. إخراج المعلومات بواسطة الأمر Msg Box.

س١٣ هل يمكن أن تستخدم الأداة نفسها للإدخال والإخراج؟ وضّح إجابتك.

نعم يمكن ذلك؛ هناك أوامر داخلية في (فيجول بيسك ستوديو) تظهر للمستخدم نافذة مصممة سابقاً من قبل الشركة المنتجة للغة البرمجة، سواء لإدخال البيانات أو لإخراج المعلومات.

س١٤ ما الذي يحدث بعد تنفيذ الإجراء التالي:

```
Dim Num As Integer, Name As String
Num=0
If Num<1 Then Name=InputBox(«أهلا بك الرجاء إدخال اسمك»)
MsgBox («أهلا بك يا»+ Name)
ENDIF
```

ستظهر نافذة صغيرة تحتوي على الرسالة التالية "أهلاً بك الرجاء إدخال اسمك" وتحتوي على مربع إدخال نقوم بكتابة الاسم فيه.

بعد إدخال الاسم وليكن مثلاً "سيف" والضغط على مفتاح الإدخال يظهر مربع الرسالة ويحتوي على العبارة "أهلاً بك يا سيف".

١٥ اكتب الأمر التالي، ولكن باستخدام جملة (Select) :

```
IF price>=1000 Then
MsgBox(«السعر غالي جدا»)
ElseIF price>=500 Then
MsgBox(«السعر غالي»)
ElseIF price>=200 Then
MsgBox(«السعر معقول»)
Else MsgBox(«السعر رخيص»)
ENDIF
```

```
SELECT CASE price
CASE price > = 1000
RESULT = Msg Box "السعر غالي جداً"
CASE 500 TO 599
RESULT = Msg Box "السعر غالي"
CASE 200 TO 499
RESULT = Msg Box "السعر معقول"
CASE ELSE
Result = Msg Box "السعر رخيص"
End SELECT
```

س١٦ لو كان لديك مصفوفة اسمها (Grades) ومخزن فيها درجات (100) من الطلاب، فما أقل درجة؟

بعد تعريف المصفوفة نقوم بكتابة الكود التالي لحساب أقل درجة:

```
Dim min

min = 0

for count = 0 to 99

if Grades (count) < Grades (count + 1)

then min = Grades (count)

next
```

اختر رمز الإجابة الصحيحة فيما يلي :

١ لكتابة برنامج هناك :

أ - ثلاث مراحل.
ب- مرحلتان.
ج- أربع مراحل.
د- خمس مراحل.

٢ تحديد عدد الواجهات والأدوات المستخدمة لكل واجهة نقصد به:

أ - تصميم الواجهات.
ب- برمجة الواجهات.
ج- تعديل الواجهات.
د- ربط الواجهات.

٣ قبل كتابة الأوامر البرمجية نحتاج إلى :

أ - تصميم الواجهات فقط.
ب- تصميم الواجهات وضبط الخصائص أولًا.
ج- ضبط الخصائص فقط.
د- كتابة خوارزم البرنامج.

٤ تصنف البيانات إلى :

أ - نوع واحد.
ب- نوعين.
ج- أربعة أنواع.
د- ثلاثة أنواع.

٥ إعطاء اسم لقيمة معينة واستخدامها داخل البرنامج هو تعريف :

أ - الثابت.
ب- المتغير.
ج- التاريخ.
د- الحروف.

٦ الجملة الصحيحة لتعريف متغير فيما يلي هي :

أ- Dim x = int .
ب- Dim 2DF As long
ج- Dim x = If .
د- Dim x As string

٧ ناتج العملية الحسابية M=2*6 + 3^2 هو:

أ- 13
ب- 20
ج- 12
د- 21

٨ العملية التي نتيجتها True فيما يلي هي :

أ- 6×4 = 5×3+4
ب- 6×4 <> 5×3+4
ج- 6×4 < 5×3+4
د- 6×4 > 5×3+4

٩ من أدوات إخراج المعلومات :

أ- RadioButton
ب- ListBox
ج- ChekBox
د- TextBox

١٠ لتنفيذ أمر معين طالما كان الشرط صحيحاً فإننا نستخدم :

أ- If .. Then .. ElseIF
ب- For.. Next
ج- Do .. While
د- Select Case