انواع البيانات

عند كتابة اجراءات في البرنامج نحتاج لتخزين واسترجاع معلومات و بيانات معينة .

هذه البيانات والمعلومات تخزن في ثوابت أو متغيرات أو مصفوفات .

الثوابت	المتغيرات	المصفوفات
مخازن يتم تعريفها أثناء التصميم حيث تأخذ قيما ثابتة ولايمكن تغيير قيمها أثناء التشغيل .	تأخذ قيماً متغيرة أثناء التشغيل .	مجموعات من المتغيرات لتخزين العديد من القيم

الأنواع الثلاثة السابقة تحتاج لتعريف نوع البيان المخزن فيها .

فهم أنواع البيانات

- المترجم لابد أن يفهم أنواع البيانات التي يتعامل معها فلابد من تحديد نوع المتغير أو الثابت الذي سيخزن فيه هذا البيان .

أنواع البيانات

نوع البيان
Boolean
byte
char
date
decimal
double
Integer
Long
single
string

تحويل البيانات من نوع الى آخر

- لن يسمح فيجوال بنقل البيانات من متغير الى آخر اذا كانوا ليسوا من نفس النوع فعملية تغيير نوع البيان تسمى بالتشكيل casting

أنواع التشكيل

لأعـــلى

- عند نقل قيمة من متغير ذو سعة اصغر الى متغير ذو سعة اكبر.

لأسفل

- عند نقل قيمة من متغير ذو سعة اكبر الى متغير ذو سعة اصغر.

عملية التحويل تتم عن طريق مجموعة من الدوال الجاهزة

الدالة	عمـــلـــها
cbool	للتحويل الى Boolean
Cbyte	للتحويل الى byte
Cint	للتحويل الى integer
Cstr	للتحويل الى string
حيث نمرر اليها القيمة المرادة (المراد تحويلها) وتعود بالقيمة بعد التحويل	

مثال

```
x = csng ( y )
```

للتحويل قيمة من نوع (y) double الى متغير من نوع (x) single

التعامل مع البيان من نوع Boolean

❖عند وضع قيمة true في متغير من نوع Boolean فإن فيجوال في الحقيقة يحفظ فيه القيمة -١ وعند وضع قيمة false فان فيجوال يحفظ فيه القيمة صفر .

عند تحويل قيمة رقمية الى Boolean فان فيجوال يفحص القيمة

❖إذا كانت صفر فانه يعتبرها false
❖إذا كانت قيمته غير الصفر فانه يعتبرها true

تعريف واستخدام الثوابت

فائدة الثوابت

- التخلص من أو تقليل أخطاء إدخال البيانات (مثال : عند استخدام الثابت c-pi) في البرنامج أكثر من مرة فانه من السهل ذكر اسم الثابت بدلا من ذكر قيمته وهي ٣.١٤١٥٩٢٦٥٣ .
- تحديث الأوامر يكون سهلا : إذا ذكرت قيمة صريحة داخل البرنامج وأردنا تغييرها فإننا نبحث عن كل مكان موجودة فيه لتغييرها أما الثابت فنغير في جملة الإعلان عنه فقط أي نغير في قيمته .
- تصبح قراءة الأوامر سهله .

تعريف واستخدام الثوابت

تعريف الثوابت

```
Const name  As  data type = value
Const c_pi  As  single = 3.141592653
```

استخدم الثوابت

```
Debug.writeline ( c_pi * 2 )
```

الاعلان عن المتغيرات (تعريفها)

Dim variable name As data type

مثال

```
Dim X As Integer
Dim I , j , k As decimal
X = 20
I = 100
J = X + 2 * I
Dim strname As string =“rana”
```

- كل نوع من أنواع البيانات له قيمة افتراضية
- فالبيان النص له قيمة " " و المتغير الرقمي له القيمة صفر

وضع القيمة الصريحة في المتغيرات

```
Strname = " dammam university "
Objbirthdate= # 7/22/1993 #
Intanswer = 42
```

استخدام المتغيرات في التعابير

```
Y = 10
X = y + 50
Z = x + y
```

- المتغيرات في فيجوال تعتبر كائنات لها خصائصها وأساليبها (دوالها) .

الاعلان الاجباري (الصريح) عن المتغيرات

- للإعلان الإجباري عن المتغيرات ننشط الخيار option explicit نجعله on في الجزء العام (قبل أي إجراء في شاشة الكود) .
- فعند استخدام متغير لم نعلن عنه مسبقا فان فيجوال يعترض بإعطاء رسالة خطأ .

مثال

```
Dim intmyvariable as integer
Intmyvariable = 10
Msgbox.show ( intmyvariable1 )
```

- في المثال السابق اذا كان جملة option explicit معطلة off فان الناتج لايعرض شئ لان جملة Msgbox.show تعرض قيمة المتغير intmyvariable1 وهذا المتغير لم نعلن عنه مسبقا ويكون محتواه فارغ لانه من نوع object، اما عند تشغيل الاعلان الاجباري on فان فيجوال يعطي رسالة خطأ و يوقف التنفيذ لان اسم المتغير intmyvariable1 لم يتم الإعلان عنه مسبقا

الالتزام الصارم بنوع البيانات

- يجعل فيجوال يجبرك على الالتزام بنوع البيانات أي لا نستطيع وضع قيمة ما في متغير فقط إذا كانت القيمة من نفس نوع البيانات الذي يتوقعه المتغير .

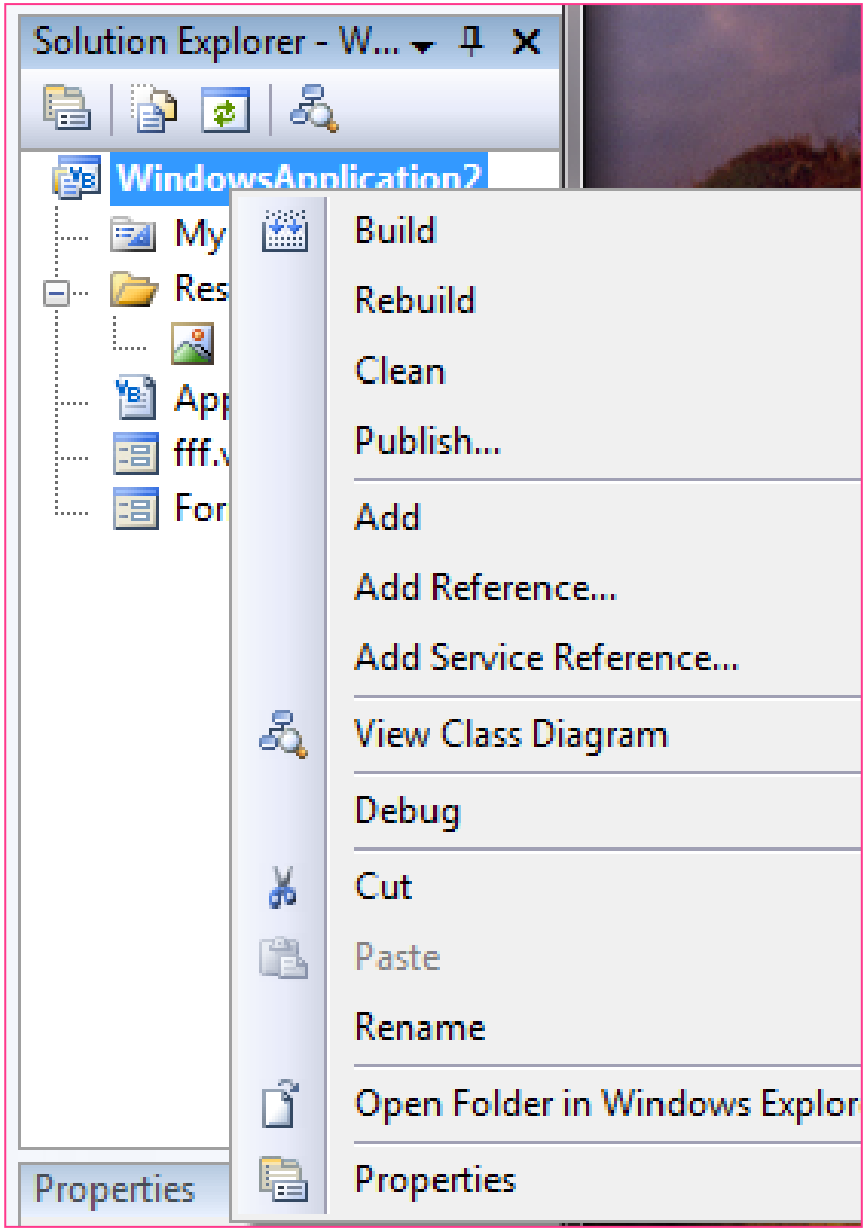
- وإذا أردنا إجراء عملية تحويل فان فيجوال لن يقوم بها تلقائيا إلا باستخدام دوال التحويل .

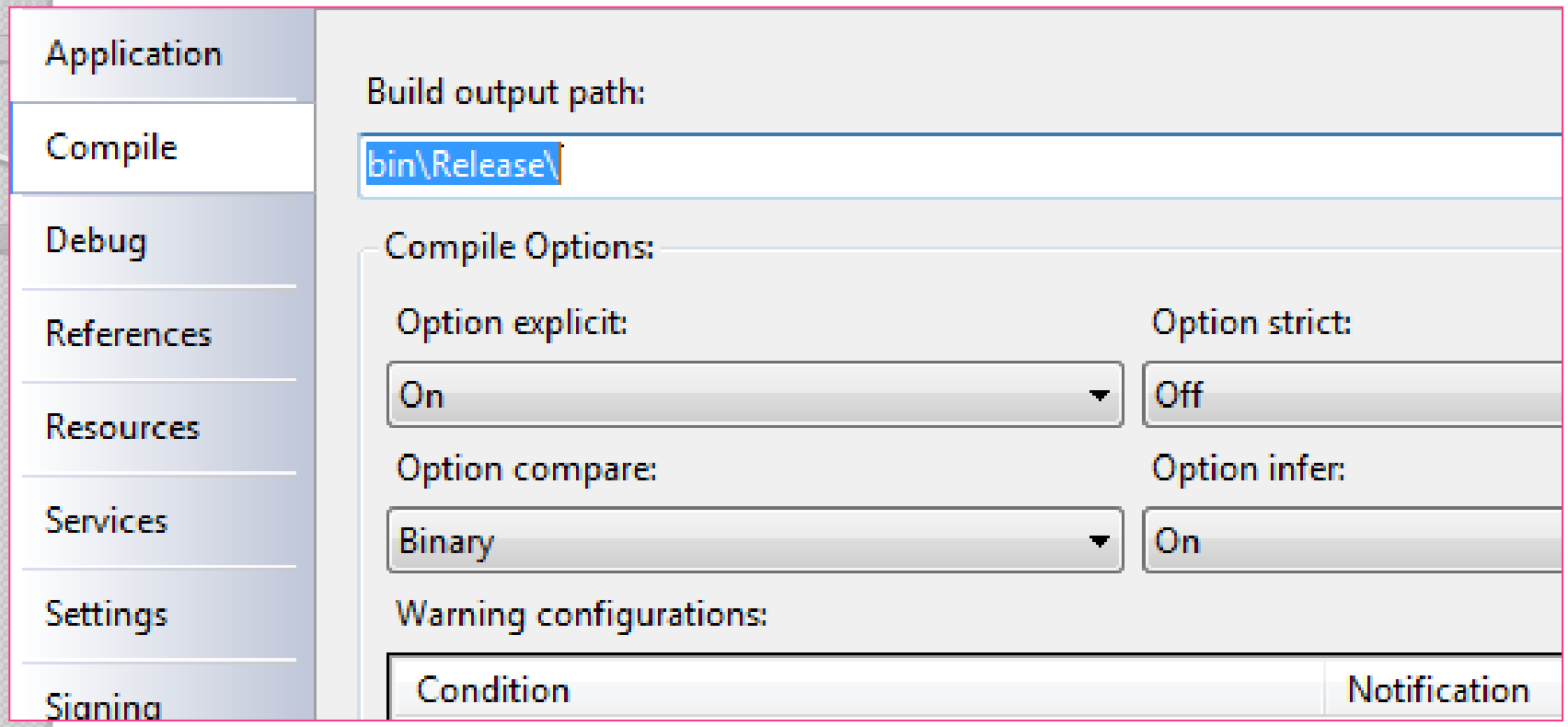
- عند إلغاء هذه الخاصية (تعطيلها) وأردنا نقل قيمة من متغير double إلى متغير single فان فيجوال سيسمح بذلك ولكنه سيلغي بعض الخانات العشرية من العدد .

لتنشيط الخاصية نتبع الآتي

- R.click على إسم المشروع في نافذة الحل ← Properties

← التبويب Compile ← Option strict ← On


Solution Explorer - W...
WindowsApplication2
My
Res
App
fff.v
For
Properties
Build
Rebuild
Clean
Publish...
Add
Add Reference...
Add Service Reference...
View Class Diagram
Debug
Cut
Paste
Rename
Open Folder in Windows Explor
Properties


Application
Compile
Debug
References
Resources
Services
Settings
Signing
Build output path:
bin\Release\
Compile Options:
Option explicit:
On
Option strict:
Off
Option compare:
Binary
Option infer:
On
Warning configurations:
Condition
Notification

تحديد المدى scope

- المدى هو المستوى الذي يمكن رؤيه الثوابت او المتغيرات أو المصفوفات فيه

أنواع المدى

- مستوى الكتلة block
- مستوى الاجراء (محلي) local
- مستوى الملف form
- مستوى العام global

مستوى الكتلة block level

- ❖ الإعلان عن متغير داخل كتلة / تركيبة من الأوامر وهذا يجعل المتغير محدودا بها ولايمكن رؤيته خارجها (مداه داخل هذه الكتلة فقط)

الكتلة

❖اوامر توجد بين عبارة افتتاحية وعبارة ختامية مثل
if oooothen .. End if ، do oooowhile .

```
If < expression > Then
        [ أوامر تنفذ إذا كان التعبير صحيح ]
End If
```

مثال

```
If  x>20  Then
        Dim Count As integer
        Count =50
        Debug.writeline (Count * 25)

End If
```

❖المتغير count يدمر فور الخروج من التركيبة if .... then وتمحى القيمة التي بداخله .

مستوى الإجراء (المحلي)

- الإعلان عن متغير داخل إجراء فان مداه يقتصر على هذا الإجراء (لا نراه خارج الإجراء) .

مستوى الملف

- أي الإعلان عن متغير داخل ملف فبذلك يمكن الإشارة إليه من داخل كافة الإجراءات الموجودة في هذا الملف .
- نعلن عن المتغيرات في قسم الإعلانات (الجزء العام) في الملف باستخدام dim .

المدى العام global

- متغير يمكن رؤيته والإشارة إليه من داخل أي إجراء في البرنامج بغض النظر عن الملف الموجود به .
- نعلن عنها في قسم الإعلانات لأحد ملفات البرمجة المستقلة modulesباستخدام كلمة public وليس ملف طبقة (شاشة لكود الخاصة بالنموذج) .

مثال ١

- الاعلان عن ثابت عام

```
Public const myconst As integer = 1
```

مثال ٢

- الاعلان عن متغير عام

```
Public strmyvariable As single = 1
```

ملاحظة هامة

- إذا أعلنا عن متغير عام في ملف طبقة (ملف نموذج) فانه سيعتبر خاصية property لهذه الطبقة ولن يكون عاما لبقية ملفات المشروع .

- لايمكن ان يكون لديك متغيران بنفس الاسم في نفس المدى .

المتغيرات الساكنة static

- عند الإعلان عن متغير داخل التركيبة أو إجراء فانه يكون موجود فقط أثناء تنفيذ هذه التركيبة / الإجراء وعند الخروج منهم يدمر هذا المتغير تماما والتخلص من قيمته وعند استدعاء هذا الإجراء مرة أخرى فانه يتم إنشاء متغير جديد تماما .

```
Public sub myprocedure ()
Dim  X As integer
X = X + 10
Debug.writeline ( X )
End sub
```

- عند استدعاء الإجراء myprocedure () فانه يطبع القيمة ١٠ على الشاشة وعند استدعاؤه مرة أخرى فانه يطبع القيمة ١٠ على الشاشة وليس ٢٠ . .

المتغيرات الساكنة static

- إذا جعلنا x متغير ساكن فانه يحتفظ بقيمته وتظل موجودة فيه ولا يمت حتى بعد الخروج من الإجراء .

```
Public sub myprocedure ()
Static X As integer
X = X + 10
Debug . writeline ( X )
End sub
```

- عند استدعاء هذا الإجراء فانه يطبع القيمة ١٠ على الشاشة وعند استدعاؤه مرة أخرى فانه يطبع ٢٠ .