**ملحق 2**

**نظام العدّ الثنائي:**

**مفهومه:**

ليس النظام العشريّ هو النظام الوحيد الذي يمكن استخدامه للعدّ، كما أنّ هذا النظام لم يكن فطريًّا كما لا بدّ وأنّك تتخيّل.. فمثلا، كانت حضارة العراق القديمة تستخدم النظام الستيني للعدّ، وذلك لملاءمته للنظام الفلكيّ، وهو ما زال مستخدمًا حتّى اليومِ في قياس الزوايا، فالدرجة تساوي 60 دقيقة والدقيقة تساوي 60 ثانية، كما أنّنا جميعا ما زلنا نستخدم النظامَ الستّينيّ يوميّا في قياس الوقت، فالساعة أيضا تساوي 60 دقيقة والدقيقة تساوي 60 ثانية، وذلك لارتباط الزوايا والوقت بحركة الأفلاك (لاحظ أنّ العلاقة بين المواقيت ونظام الفلك هي علاقة متوقّعة، فنحن نقيس الوقت تبعا لحركة الأرض حول نفسها وحول الشمس).

وكقاعدة: يمكنك ابتداع أي نظام عدّ تريده، حتّى ولو كان النظام المئوي أو الألفي، ولكنّ المشكلة التي ستواجهك هي: من أين تأتي برموز لهذا النظام لاستخدامها كأرقام؟

ففي النظام العشريّ، نستخدم عشرة رموز كأرقام (0، 1، 2، ...... ، 9) ، بحيث يمكن توليد أعداد لا نهائيّة منها بتوليفها معا، مثل 123، 7638، 649863.. إلخ.

(كان للوح الحساب الصينيّ الفضل في ابتداع هذا النظام، وعنهم نقله الهنود، وعنهم أخذه العرب حيث قنّنوا قواعد هذا النظام وابتكروا رمزًا للصفر).

وفي هذا النظام أيضا، يكتسب الرقم قيمته تبعًا للخانة التي يوجد فيها، ففي خانة الآحاد تكون قيمته كما هي، وفي خانة العشرات تكون قيمته مضروبة في 10، وفي خانة المئات تكون قيمته مضروبة في 100، وهلم جرا.... .

فمثلا: العدد 111 هو عبارة عن 1 (في الآحاد) + 10 (1 × 10 لأنّه في خانة العشرات) + 100 (1× 100 لأنّه في خانة المئات).. وهكذا ترى أنّ الرقم واحد قد اختلفت قيمته على حسب موضعه في العدد.

**إنّ لهذه الحقيقة صيغةً عامّة:**

في النظام العشريّ، لو كان لديك رقم "ر"، فإنّ قيمته "ق" بالنسبة لموقعه "م" في العدد هي:

**ق = ر × 10 (م -1).**

 فمثلا الرقم 5 (ر =5) في خانة الآحاد (م = 1) قيمته:

ق = 5 × 10 (1 -1) = 5 × 1 = 5.

وفي خانة العشرات (م = 2) قيمته:

ق = 5 × 10 (2 -1) = 5 × 10 = 50.

وفي خانة العشرات (م = 2) قيمته:

ق = 5 × 10 (3 -1) = 5 × 100 = 500... وهكذا.

ولحساب العدد ككل، يتم جمع القيمة الموضعيّة لكل رقم من أرقامه.. فمثلا:

العدد 143 = 3 × 10 (0) + 4 × 10 (1) + 1 × 10 (2) = 3 + 40 + 100.

طبعا ستتساءل في استياء: لماذا تكرّر لنا الكلام الذي درسناه في الصف الأوّل الابتدائيّ؟

ولو صبرت قليلا، لعرفت مدى أهمّيّة هذه القواعد "الطفوليّة"، فهذه القواعد يمكن تعميمها على أيّ نظام.. فبدلا من الأساس 10 في المعادلات السابقة، ضع بدلا منه أساس النظام الذي تريده.. جرّب النظام الخماسي (0، 1، 2، 3، 4)، والنظام العشريني (0، 1، 2، ....... ، 19)، وأي نظام آخر تريده.

ونظرًا لأن الكمبيوتر يعمل بدوائر كهربيّة دقيقة، يمكنها أن تدرك حالتين فقط: مرور التيار (وسنرمز له رياضيّا بالرمز 1)، أو انقطاعه ( وسنرمز له رياضيّا بالرمز 0)، فإنّ النظام الثنائيّ (0، 1) ، هو النظام الأمثل بالنسبة للكمبيوتر، وهو هنا ما يعنينا.

ورغم أنّه سيُشعرك بالغرابة، إلا إنّه يسير على نفس قواعد النظام العشريّ في العدّ والعمليات الحسابيّة (الجمع والطرح والضرب والقسمة).

وفي الأمثلة التالية، سنضع العدد بين قوسين، ونضع أساس النظام بجوارهما، لنرى كيف يمكن التحويل بين النظام العشري والنظام الثنائي:

(1)10 = (1)2.

(2)10 = 0 × 2 0 + 1 × 2 1 = (10)2.

(3)10 = 1 + 2 = 1 × 2 0 + 1 × 2 1 = (11)2.

(11001)2 = 1 × 2 (0) + 0 × 2 (1) + 0 × 2 (2) + 1 × 2 (3) + 1× 2 (4)

= 1 + 0 + 0 + 8 + 16 = (25)10.

**أقصى عدد عشري يقبله عدد من الخانات في النظام الثنائي:**

أقصى عدد يمكن أن تضعه في 3 خانات عشريّة هو 999، أي 1000 - 1، أي 10 3 – 1، وأقصى عدد يمكن أن تضعه في أربع خانات هو 10 4 – 1 = 10000 – 1 = 9999، وأقصى عدد تضعه في (ن) من الخانات هو  10 ن – 1.

بالمثل، فإنّ أقصى عدد يمكن وضعه في خانتين ثنائيّتين هو (11)2 = 3 = 2 2 – 1، وأقصى عدد يمكن أن تضعه في 8 خانات هو  2 8 – 1 = 256 -1 = 255 (هذا هو السبب في أن المتغيّر من النوع "وحدة الذاكرة" Byte يقبل الأعداد من 0 حتّى 255).. وعموما: أقصى عدد تضعه في (ن) من الخانات هو  2 ن – 1.

**الأعداد الموجبة والسالبة:**

يمكن تمثيل الأعداد السالبة في النظام الثنائيّ، بتخصيص أقصى خانة على اليسار للإشارة، فلو كانت قيمتها صفرا كان العدد موجبا، ولو كانت قيمتها 1، كان العدد سالبا.. أي أنّ رقما مثل (11001)2 يمكن أن يمثّل الرقمين العشريين 25 أو -9.. فكيف يمكن التفريق بينهما؟

إنّ ذلك يرجع لطريقة تعاملك أنت مع العدد، ولو كنّا نستخدم إحدى لغات البرمجة منخفضة المستوى Low Level، لكنت أريتك كيف تُخبر الكمبيوتر بمراعاة الإشارة أو عدم مراعاتها.. ولكن كل هذه المتاهات لا تهمّنا هنا، فأنت تتعامل مع أعلى لغات البرمجة: لغة VB، وهي تتولّى عنك كل العمليات المعقّدة.

يكفيك هنا أن تتذكّر، أنّ المتغيّر من النوع Byte يحمل فقط أعدادا موجبة، لهذا يُكتب العدد في 8 خانات، أي أنّ أكبر عدد يستوعبه هو 255.. أمّا المتغير من النوع SByte، فيمكن أن يحتوي على أعداد سالبة، ممّا يعنى أنّ هناك خانة مخصّصة للإشارة، وبهذا يُكتب العدد في 7 خانات فقط، أي أنّ أكبر عدد يستوعبه هو 2 7 – 1 = 127.

أهمّ ما يجبُ أن تلاحظه هنا، هو أنّ هذه هي الطريقة التي يتعامل بها الكمبيوتر مع كلّ شيء.. إنّه لا يعرف غير الأرقام، ولكنّه يحوّلها إلى مدلولات أخرى على حسب رغباتنا نحن.. إنّ 8 خانات ثنائيّة تحتوي كلّها على الرقم 1 (11111111)2 تعطينا الرقم 255 لو اعتبرناها موجبة، و – 127 لو اعتبرناها سالبة، واللون الأحمر لو اعتبرناها لونا، والرمز "?"  لو اعتبرناها حرفا، ونغمة ما لو اعتبرناها صوتا..... وهكذا.. كلّ ما يفعله الكمبيوتر هو أن ينظر في جداول التحويل المناسبة، ويعرض لنا القيمة الناظرة.. إنّ كلّ شيءٍ بالنسبةِ للكمبيوتر نسبيّ، فالأرقام المجرّدة التي يتعامل معها، يمكن أن تفعل كلّ شيء.. المهم: هو الزاوية التي تنظر لها منها.

**العمليّات الحسابيّة:**

لو حاولت إجراء عمليات الجمع على الأعداد الثنائيّة، فستجد أنّ الأمر في منتهى البساطة، فأنت تجمع رقمي خانتي الآحاد في العددين، وتحمل معك الباقي لتجمعه على ناتج جمع رقمي خانتي الثنائيات، ثم خانتي الرباعيات، وهكذا... .. نفس ما تفعله في النظام العشريّ، وإن كنت هنا ستصادف ثلاث حالات فحسب:

1- أن يكون الرقمان صفرين، فيكون مجموعهما صفرا.

2- أن يكون أحد الرقمين 1 والآخر صفرا، فيكون مجموعهما 1.

3- أن يكون كلا الرقمين 1، فيكون مجموعهما 0، والباقي واحد.

طبعا الحالة الأخيرة هي التي ستُربكك.. نفس ما حدث لك عندما تعلّمت أن جمع 9 + 1 = 0 والباقي 1 (أي 10).. بالمثل، فإنّ 1+1 = (2)10 = (10)2.. كل ما يحدث هو أنّك عندما تتجاوز أقصى رقم في النظام (9 في العشري ، و 1 في الثنائي)، تنتقل إلى الخانة التالية ومعك الباقي!

فلنرَ مثالا للجمع:

1000111

+  1100

ــــــ

1010011

وبالمثل، يمكن إجراء باقي العمليات الحسابيّة، كالطرح والضرب والقسمة، فهي كما قلنا، مماثلة تماما لمثيلاتها في النظام العشريّ.

ولكنّنا لن نركّز على العمليات الحسابيّة، لأنّ التطبيقات التي تبنيها لا تستخدم الحسابات الثنائيّة، وإنّما تستخدم النظام العشريّ.

إنّ الذي يهمّنا هنا هو العمليّات المنطقيّة، فكثيرا ما ستحتاجها في برامجك.

**العمليّات المنطقيّة**

**"ليس" NOT:**

بالنسبة للمتغيّرات المنطقيّة، فإنّ هذه العمليّة تعكس قيمة المتغيّر الذي تُجرى عليه، من صواب لخطإ والعكس.. والجملة التالية تُريك مثالا لهذا:

**Failure = Not Failure**

وهي طريقة مختصرة للطريقة التالية:

**If Failure = True Then**

**Failure = False**

**Else**

**Failure = True**

**End If**

وليست هذه هي الوظيفة الوحيدة لعملية "ليس" NOT، إذ يمكن استخدامها مع الأرقام الثنائيّة، لعكس كل صفر إلى واحد، وكل واحد إلى صفر (يسمى هذا Bitwise Operation)، بحيث لو جمعت العدد ومعكوسه، ينتج عدد تتكوّن كل خاناته من الرقم 1.. فلو جربت هذا المثال:

**Dim X As Byte = 5**

**MsgBox (NOT X)**

لظهرت لك رسالة تعرض الرقم 250.. هل أدهشك الرقم؟

في البداية تعالَ نمثّل الرقم 5 بالطريقة الثنائيّة.

(5)10 = 1 + 4 = 1 × 2 0 + 0 × 2 1 + 1 × 2 2 = (101)2.

وبهذا يكون المعكوس هو (010)2 = (2)10.

فمن أين أتت القيمة 250 إذن؟

إنّ المتغيّر Byte يُمثّل في الذاكرة بثمانية خانات، لهذا يجب أن نضع خمسة أصفار على يسار التحويل السابق.. وكما في النظام العشريّ، لا يؤثّر وضع صفر على اليسار في قيمة الرقم، ولكن عند عكس الرقم، سيتحوّل كلّ صفر إلى 1، ممّا سيكون له أكبر الأثر:

(5)10 = (00000101)2.

وبهذا يكون المعكوس هو (11111010)2 = 2 + 8 + 16 + 32 + 64 + 128 = (250)10.

ولو جمعنا الرقم ومعكوسه، لنتج لنا الرقم (11111111)2، الذي يمثّل أقصى عدد يمكن أن يحمله المتغيّر من النوع Byte، وهو 255.

إذن يمكن استنتاج ما يلي:

**مكوس المتغير = أقصى عدد يمكن أن يحمله المتغيّر – قيمة المتغيّر الحاليّة.**

وبناء على هذا، كان من الممكن حساب ناتج العملية السابقة مباشرة، دون التحويل بين النظامين العشريّ والثنائيّ، وذلك كالتالي:

ليس (5) = 255 – 5 = 250.

لهذا يسمى معكوس الرقم بـ "مرافق الرقم" Complement.

**"و" AND:**

تستخدم هذه العملية مع  متغيّرين منطقيّين، للتأكّد من حدوث كليهما معا: الأوّل **(و**) الثاني.. خذ هذا المثالَ للتوضيح:

 افترض أنّك تريد فتح ملف.. يجب أن تتأكّد أولا أنّه ليس للقراءة فقطReadOnly وليس مُخفىً Hidden:

**If ReadOnly = False AND Hidden = False Then**

**{ الكود المناسب لفتح الملف }**

**Else**

**MsgBox("لا يمكن فتح الملف إذا كان للقراءة فقط أو مُخفى")**

**End If**

لاحظّ أن And لا تربط فقط بين متغيرين، بل يمكن أن تربط كذلك بين شرطين، ففي المثال السابق، ReadOnly = False هي جملة شرط، و Hidden = False هي جملة شرط أخرى.

**ولكن ماذا عن استخدام AND مع الأعداد الثنائيّة؟**

الجدول الآتي يوضّح لنا وظيفة هذه العمليّة عند تطبيقها على خانتين:

|  |  |  |
| --- | --- | --- |
| **أ** | **ب** | **أ و ب** |
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

أي أنّ ناتج عملية "و" على رقمين في خانتين متناظرتين من عددين، هو صفر دائما، إلا إذا كان كلا الرقمين معا 1.. جرّب هذا المثال:

**Dim A As Byte = 5 , B As Byte = 34**

**MsgBox (A AND B)**

في البداية تعال نحول العددين للنظام الثنائي:

5 = (00000101)2.

34 = 2 + 32 = 0 × 2 0 + 1 × 2 1 + 0 × 2 2 +  0 × 2 3 + 0 × 2 4 + 1 × 2 5 = (00100010)2.

تلاحظ أنّ الرقم ا موجود في العدد الأوّل في الخانتين الأولى والثالثة فقط، بينما يناظره الرقم 0 في هاتين الخانتين في العدد الثاني.. إذن كل الخانات الناتجة عن عملية "و" ستكون أصفارا.. لهذا ستظهر لك الرسالة وعليها الرقم 0.

**وأهمّ خواص هذه العمليّة:**

1- أ و أ = أ.

2- أ و 0 = 0.

3- أ و 1 = أ.

4- أ و ليس أ = 0 (لا يمكن أن يجتمع الشيء وعكسه.. هذا شرط خاطئ دائما).

**"أو" OR:**

تستخدم هذه العملية مع  متغيّرين منطقيّين، للتأكّد من حدوث أحدهما على الأقل: الأوّل **(أو)** الثاني (**أو**) كليهما معا.. خذ هذا المثالَ للتوضيح:

**If ReadOnly Or Hidden Then**

**MsgBox("لا يمكن فتح الملف إذا كان للقراءة فقط أو مُخفى")**

**Else ' الملف ليس مخفى وليس للقراءة فقط**

**{ الكود المناسب لفتح الملف }**

**End If**

ويمكن إيجاد صيغة أخرى للكود السابق، باستخدام NOT، فنقوم بفتح الملفّ إذا **لم** يكن (للقراءة فقط، **أو** مخفى):

**If Not (ReadOnly or Hidden) Then**

**{ الكود المناسب لفتح الملف }**

**Else**

**MsgBox("لا يمكن فتح الملف إذا كان للقراءة فقط أو مُخفى")**

**End If**

لاحظ أنّ النفي يشمل ناتج كل ما بين القوسين، فمثلا لو كان الملف للقراءة فقط ReadOnly = True، لكان ناتج عملية "أو" هو True، حيث ستتم عليه عملية النفي NOT، ليصبح الناتج النهائيّ False.. إذن فلن يتمّ تنفيذ جملة الشرط، لأنّ الشرط خطأ، وسينفّذ البرنامج الحالة الأخرى Else.

وما زالت هناك طريقة ثالثة لتنفيذ نفس البرنامج، حيث يمكننا أن نفكّ القوس، بإدخال NOT على حدّي عملية Or مع عكس المعامل Or نفسه.. ولكن ما هو عكس Or؟.. تعالَ نستنتجها:

لو قلت: **لم** يأت (محمد **أو** أحمد)، فأنت تعني أن الاثنين معا لم يأتيا بعد.. ويمكن أن تقول هذه الجملة بطريقة أخري كالتالي: **لم** يأتِ محمد **و** **لم** يأت أحمد.

إذن فنفي OR هو AND، كما أنّ نفي AND هو Or.

إذن: **ليس** (للقراءة فقط **أو** مخفى) = **ليس** للقراءة فقط **وليس** مخفى:

**If Not ReadOnly And Not Hidden Then**

**{ الكود المناسب لفتح الملف }**

**Else**

**MsgBox("لا يمكن فتح الملف إذا كان للقراءة فقط أو مُخفى")**

**End If**

**ولكن ماذا عن استخدام OR مع الأعداد الثنائيّة؟**

الجدول الآتي يوضّح لنا وظيفة هذه العمليّة عند تطبيقها على خانتين:

|  |  |  |
| --- | --- | --- |
| **أ** | **ب** | **أ أو ب** |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

 أي أنّ ناتج عملية "أو" على رقمين في خانتين متناظرتين من عددين، هو 1 دائما، إلا إذا كان كلا الرقمين معا صفرا.. جرّب هذا المثال:

**Dim A As Byte = 5 , B As Byte = 34**

**MsgBox (A Or B)**

5 =   (00000101)2.

34 = (00100010)2.

والناتج = (00100111)2 = 1 + 2 + 4+ 32 = 39.

وهو الرقم الذي سيعرضه لك الرسالة.

تلاحظ أن الناتج 39 = 5 + 34.. لا تجعل ذلك يخدعك، فعملية "أو" ليست عملية جمع، والذي جعلها تبدو كذلك هنا، هو أنّ موضع الرقم 1 غير متناظر في العددين، ولو نفّذت الكود التالي لفهمت ما أعنيه:

**Dim A As Byte = 6 , B As Byte = 34**

**MsgBox (A OR B)**

حيث ستجد أن الرسالة تعرض لك الرقم 38، بينما أنّ ناتج جمع ( 6 + 34) هو 40.

**وأهمّ خواص هذه العمليّة:**

ا- أ أو أ = أ.

2- أ أو 0 = أ.

3- أ أو 1 = 1.

4 – أ أو ليس أ = 1 (الشيء أو عكسه.. هذا شرط صحيح دائما).

**"أو المقصورة" XOR:**

إن العمليات الثلاث السابقة NOT و OR و AND هي العمليّات الأساسيّة التي يمكن عن طريقها اشتقاق أيّ عمليّة منطقيّة لأخرى.. بل ومنها يمكن اشتقاق كلّ العمليات الحسابيّة (كالجمع والطرح والضرب والقسمة)، وأيّ دالة أخرى.. وهذه هي الطريقة التي يعمل بها الكمبيوتر، حيث يتمّ تمثيل كلّ الوظائف باستخدام هذه العمليّات المنطقيّة.. ولكنّ شرح هذا خارج عن نطاقنا الآن.

ومن العمليّات التي يمكن اشتقاقها، وتعتبر حيويّة للغاية، العمليّة XOR، ووظيفتها كالتالي:

A XOR B = (A AND NOT B) OR (B AND NOT A).

أ XOR ب = (أ وليس ب) أو (ب وليس أ).

والجدول التالي يبيّن وظيفتها:

|  |  |  |
| --- | --- | --- |
| **أ** | **ب** | **أ XOR ب** |
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 0 |

**وأهمّ خواص هذه العمليّة:**

1- أ XOR أ = 0.

2- أ XOR 0 = أ.

3- أ XOR 1 = ليس أ   (A XOR 1 = NOT A).

4- أ XOR ليس أ = 1.

**تكوين الأقنعة Masking**

 **المؤشّرات Flags والأقنعة Masks:**

كثيرا ما نستخدم متغيرا واحدا لتخزين معلوماتٍ عن حالة العديد من الأشياء، إذا كانت هذه الحالة لها قيمتان فحسب (نعم / لا.. 1/0).

فمثلا، يمكن أن تخزّن في متغيّر واحد حالة 8 طلاب، من حيث الغياب والحضور.. كيف تفعل ذلك؟

بسيطة: عرّف متغيّرا من النوع Byte وليكن Students.. أنت تعرف أنّه يحجز وحدة ذاكرة، أي أنّه يتكوّن من 8 خانات.. سنخصّص لكل طالب خانة منها، والطالب الموجود سنضع في خانته الرقم 1، والطالب الغائب سنضع في خانته الرقم 0.. وبهذا سيحتوي المتغيّر على عدد يدلّ على حالة الطلاب الثمانية.

لاحظ أنّ هذه التقنية ستؤدّي لما يلي:

لو كان كلّ الطلاب غائبين، فسيحتوي المتغيّر على العدد 0 .

لو كان الطالب الأوّل موجودا بمفرده، فسيحتوي المتغيّر على العدد 1.

لو كان الطالب الثاني موجودا بمفرده، فسيحتوي المتغيّر على العدد 2.

لو كان الطالب الثالث موجودا بمفرده، فسيحتوي المتغيّر على العدد 4.

لو كان الطالب الرابع موجودا بمفرده، فسيحتوي المتغيّر على العدد 8.

لو كان الطالب الخامس موجودا بمفرده، فسيحتوي المتغيّر على العدد 16.

لو كان الطالب السادس موجودا بمفرده، فسيحتوي المتغيّر على العدد 32.

لو كان الطالب السابع موجودا بمفرده، فسيحتوي المتغيّر على العدد 64.

لو كان الطالب الثامن موجودا بمفرده، فسيحتوي المتغيّر على العدد 128.

هل لاحظت شيئا؟

لو كان رقم الطالب ر، فإنّ تأثيره على المتغيّر يساوي (2) ر-1.. واضح إذن أنّنا نستخدم النظام الثنائيّ، وليس العشريّ.

لاحظ كذلك أنّ وجود الطالب الأوّل والثالث يعطي للمتغيّر القيمة 1 + 4 = 5، وهي قيمة يستحيل أن تنتج عن جمع طالبين آخرين غيرهما.. أليست نتيجةً شيّقة؟!

إنّ هذه النتيجة تعني ببساطة، أنّ كلّ عدد ناتج عن تباديل وتوافيق حضور وغياب الطلاب، هو عدد متفرّد، يمكن من خلاله تحديد حالة الطلاب بلا أيّ لبس.

إنّ المتغيّر في هذه الحالة يُسمّى مؤشّرا Flag، ويمكنك أن تعرف كيفيّة تعريف المؤشّرات، ضمن موضوع تعريف المرقّمات Enums في [نهاية الفصل الثالث](file:///C:\\01-%20%D8%A3%D8%B3%D8%A7%D8%B3%D9%8A%D9%91%D8%A7%D8%AA%20%D8%A7%D9%84%D9%84%D8%BA%D8%A9\\03-%20%D8%A7%D9%84%D9%85%D8%AA%D8%BA%D9%8A%D8%B1%D8%A7%D8%AA\\15-%20%D8%A7%D9%84%D8%A3%D9%86%D9%88%D8%A7%D8%B9%20%D8%A7%D9%84%D8%AA%D9%8A%20%D9%8A%D9%8F%D8%B9%D8%B1%D9%91%D9%81%D9%87%D8%A7%20%D8%A7%D9%84%D9%85%D8%B3%D8%AA%D8%AE%D8%AF%D9%85\\1-%20%D8%A7%D9%84%D9%85%D9%8F%D8%B1%D9%82%D9%91%D9%8E%D9%85%D9%8E%D8%A7%D8%AA.htm).

ولكن.. ما هو موضوع الأقنعة هذا؟

الفكرة هنا هي كيف نغيّر حالة أحد الطلاب دونَ أن نؤثّر على باقي الطلاب؟

افترض مثلا، أنّك تريد وضع . في خانة الطالب الخامس، لأنّك اكتشفت أنّه غائب.. أوّل ما سيجول بذهنك، هو أن تكتب جملة Select Case تختبر فيها 255 حالة للمتغيّر Students، هي كل تباديل وتوافيق حضور وغياب الطلاب الثمانية.. طبعا هذا حل غير عمليّ.. الأفضل أن تبحث عن وسيلة تسمح لك بوضع صفر في الخانة الخامسة من العدد دون تغيير قيم باقي الخانات.. هذه الحركة تسمّى إلباس القناع Masking، حيث يبدو الأمر وكأنّك ألبست العدد قناعًا حتّى لا يظهر منه سوى الخانة الخامسة فقط.

ولكن كيف نصنع هذا القناع Mask؟

لدينا هنا ثلاثة أقنعة، كلّ منها له وظيفة محدّدة.. تعال نتعرّف عليها، ونرى كيف نصنعها.

**تكوين قناع "و":**

في البداية سنفترض أنّ كل الطلاب موجودون، لهذا سنبدأ المتغيّر بالقيمة 255 (كل الخانات بها 1).. والآن نريد تسجيل غياب الطالب الخامس.. هنا يأتي دور الخاصيتين الثانية والثالثة لعملية And:

-       فعند إجرائها بين أيّ رقم وبين الواحد، يظلّ الرقم كما هو:

X And 1 = X

-       أمّا لو أجريناها بين أيّ رقم والصفر، فسيكون الناتج صفرا.

X And 0 = 0

إذن فسيكون قناعنا عبارة عن عدد، كل خاناته تحتوي على الرقم 1، ما عدا الخانة الخامسة، فستحتوي على 0:

(11101111)2 = 255 – 16 = 239.

إذن فالكود سيكون كالتالي:

**Dim Students As Byte = 255**

**Students = Students AND 239**

وبهذه الطريقة، ستجد أنّك تستطيع تأليف 8 أقنعة فقط، تعبر بها عن غياب كل طالب من الطلبة.. حاول أن تدرّب نفسَك على تكوينِ هذه الأقنعة.. ويمكنك تسهيل الأمر على نفسك، لو لاحظت أنّ الرقم (11101111)2 هو معكوس الرقم ((00010000)2 أي 16.. معنى هذا أنّ باستطاعتك التعبير عن الكود السابق كالتالي:

**Students = Students And (Not 16)**

ولو لاحظت أن 16 = (2)5-1، فيمكنك أن تفهم كيف يمكن أن تعبّر الجملة التالية عن غياب الطالب رقم N:

**Students = Students And (Not 2^(N-1))**

وطبعا يجبُ عليك تعريف المتغيّر N ومنحه قيمة أولا.

**قاعدة:**

استخدم قناع And لجعل قيمة خانة أو أكثر في عدد صفرًا، دون تغيير باقي خانات العدد.. ويتم تكوينه بوضع صفر في القناع في المواضع المناظرة للخانات التي تريد تغييرها، ووضع 1 في باقي خانات القناع.

**تكوين قناع Or:**

افترض الآن أنّ الطالب الخامس الذي سجّلته في الغياب بقناع And قد حضر متأخّرا، وتريد إثبات حضوره مرّةً أخرى، وكان هناك عدد آخر من الطلاب الغائبين، بحيث لا تريد تغيير باقي قيم الخانات.. في هذه الحلة لن يفيدنا قناع And، لأنّه لا يُستخدم إلا لوضع الأصفار.. هنا يمكن استخدام قناع Or، بتكوين رقم كله أصفار (الخاصيّة الثانية: الصفر لا يغيّر الخانة المناظرة في عملية Or)، ما عدا الخانة الخامسة فسنجعلها 1 (الخاصيّة الثالثة).. في هذه الحالة سيكون القناع هو (00010000)2 = 16.

**Students = Students Or 16**

وبصورة عامّة، يمكنك تغيير الخانة رقم N لتصبح 1، بالجملة التالية:

**Students = Students Or 2^(N-1)**

**قاعدة:**

استخدم قناع Or لوضع خانة أو أكثر في عدد بواحد، دون تغيير باقي خانات العدد.. ويتم تكوينه بوضع 1 في القناع في المواضع المناظرة للخانات التي تريد تغييرها، ووضع 0 في باقي خانات القناع.

**تكوين قناع XOR:**

تقدّم لنا XOR نوعا هامّا جدا من أنواع الأقنعة، وذلك في حالة إذا ما أردنا عكس قيمة بعض الخانات في عدد، دون تغيير باقي الخانات.

افرض مثلا ـ مثلا يعني!! ـ أنّك تريد أن تعبث مع الطالب الخامس الذي أرهقنا مع قناعي And و Or، فقرّرت أن تعكس حالات حضوره هذا الأسبوع كنوعٍ من المقالب الساخنة.. في هذه الحالة ستستخدم XOR مع نفس قناع Or (لا تنس أن "أو المقصورة" XOR هي حالة خاصة من Or)، لأنّ الخانة التي بها واحد في القناع تعكس قيمة الخانة التي تناظرها في العدد، بينما لا تؤثّر الخانة التي بها صفر فيما يناظرها من خانات.

**Students = Students XOR 16**

لا ترتبك لأنّنا استخدمنا الرقم 16 مرتين: مرة مع OR ومرة مع XOR، فالناتج سيختلف في الحالتين، لأنّ العبرة ليست بالقناع فقط، وإنما بالعمليّة كذلك.

**قاعدة:**

استخدم قناع "أو المقصورة" XOR لعكس قيمة خانة أو أكثر في عدد، دون تغيير باقي خانات العدد.. ويتم تكوين هذا القناع بوضع 1 في القناع في الخانات المناظرة للخانات التي تريد عكسها في العدد، ووضع 0 في باقي خانات القناع.

**مشروع الأقنعة:**

طبعا ستتساءل: ولماذا هذا اللفّ والدوران؟.. لماذا لا أعرّف 8 متغيرات: واحد لكل طالب، بدلا من موضوع المؤشّرات والأقنعة هذا؟

معك حق، ولكن أحيانا تبدو الطريقة الأولى أوفر في الحجم، خاصّة أنّك لن تتعامل مع 8 طلاب فقط في كلّ مرّة.

عامّةً، هذه مجرّد طريقة، تستطيع أن تضيفها لسجلّ خبراتك، وهي شائعة الاستخدامِ في البرمجة، وستقابلك كثيرًا.

فمثلا، عند التعاملِ مع الملفّات، هناك خاصيّة اسمها GetAttributes، تعيد لك سمات الملفّ (للقراءة فقطReadOnly ، مخفى Hidden، طبيعيّ Normal... إلخ.).. المشكلة أنّ الملفّ قد يمتلك أكثر من سمة في نفس الوقت، كأن يكونَ للقراءة فقط وخفيّا.. فكيف إذن يمكن التعبير عن كلّ هذه السمات بخاصيّة واحدة؟.. آه.. فهمت!.. سنستخدم المؤشّرات والأقنعة!

تعالَ نرى بعض الكود، والذي ستجده مشروحا بالتفصيل في الفصل العاشر "الملفّات".

في البداية يجب أن تكتب جملة الاستيراد التالية في أقصى بداية كود النموذج، قبل جملة تعريف النموذج نفسه:

**Imports System.IO**

والآن تعالَ نقرأ سمات الملفّ "C:\boot.ini":

**Dim FA As FileAttributes = File.GetAttributes("C:\boot.ini")**

**MsgBox(FA.ToString)**

ثمّ تعالَ نرى استخداما لكلّ نوع من الأقنعة:

إنّ الكود التالي يستخدم قناع AND للتخلّص من سمة خفيّ Hidden، ويعيد الملفّ قابلا للقراءة والكتابة.. لاحظ أنّنا لن نستخدم الأرقام مباشرةً، بل سنستفيد من المرقّمات، التي تجعل الكود واضحا ومفهوما:

**FA = FA And (Not FileAttributes.Hidden)**

**MsgBox(FA.ToString)**

والكود التالي يضيف سمة للقراءة فقط لسمات الملفّ:

**FA = FA Or FileAttributes.ReadOnly**

**MsgBox(FA.ToString)**

أمّا الكود التالي فيعكس حالة السمة System ثمّ يعكسها مرّة أخرى لتعود لحالتها الأولى:

**FA = FA XOr FileAttributes.System**

**MsgBox(FA.ToString)**

**FA = FA XOr FileAttributes.System**

**MsgBox(FA.ToString)**

ويمكنك تجربة هذا الكود في تطبيق Masking في مجلّد برامج هذا الملحق.