Tự học VBA – Bài 1: Thanh công cụ Developer trong Excel

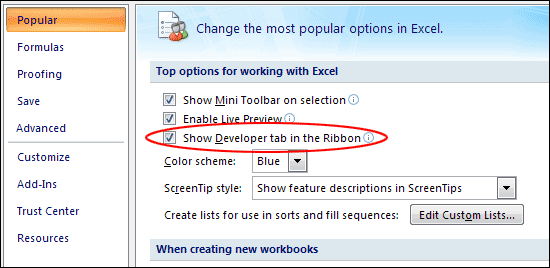
Trước khi bắt đầu làm một cái gì đó, chúng ta cũng cần phải có sự chuẩn bị để công việc được thuận lợi, giả dụ như trước khi chinh phục các cô gái chúng ta cần chuẩn bị một cái ví Thạch Sanh ^\_\_^ . VBA cũng vậy, trước khi vào học, các bạn cần phải **làm hiển thị** thanh công cụ Developer (Nhà phát triển). Nói một cách hình ảnh thì thanh công cụ **Developer**  như cái “ví Thạch Sanh” giúp bạn có thể cưa đổ cô nàng VBA đỏng đảnh này. Nhưng hãy tin tôi đi, khi có được sự phục vụ của nàng, bạn sẽ thực sự thăng hoa trong công việc của mình.

Hãy để ý từ “**làm hiển thị**” tức là mặc định Developer  không hiển thị trên thanh công cụ phía trên cùng của Excel.  Chúng ta sẽ làm theo các bước sau đây để hiển thị nhé.

**DÀNH CHO CÁC BẠN ĐANG DÙNG EXCEL 2007**

Các bạn click vào biểu tượng Office hình tròn, góc trên, bên phải –> Click Excel Option ở phía dưới –> Khi bạn thấy hộp thoại Option hiện ra –> Click vào thẻ Popular ở bên trái hộp thoại

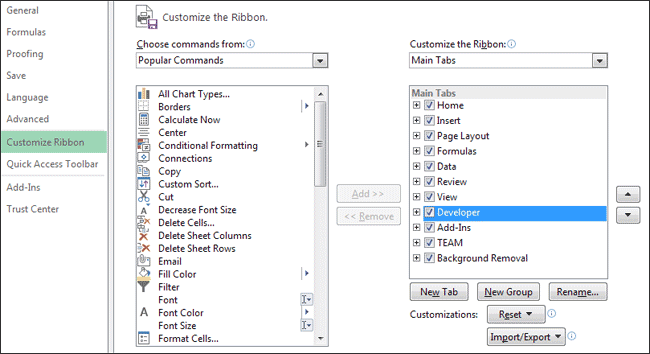
Tại khu vực “Top options for working with Excel”  các bạn check chọn “Show Developer tab in the Ribbon” là xong.



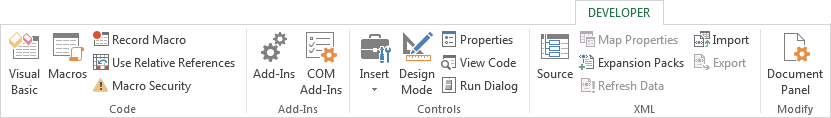
**DÀNH CHO CÁC BẠN ĐANG DÙNG EXCEL 2010 hoặc 2013**

Đối với hai phiên bản Excel này, các bạn làm chung một cách tuần tự như sau:

Click vào menu **File** (góc trái, bên trên) –> Click **Options** –> Từ hộp thoại hiện ra, click tab **Customize Ribbon** ở bên trái hộp thoại. Từ bên phía phải,  bạn sẽ nhìn thấy được một khu vực gọi là “**Customize the Ribbon**“. Dưới  mục “Main Tabs” check chọn **Developer**:



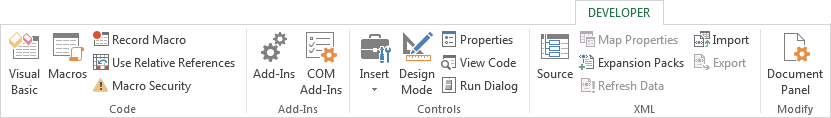
Khi bạn hoàn thành các bước trên, Bạn sẽ nhìn thấy Developer ở thanh công cụ phía trên của Excel (Hình dưới đây là sự xuất hiện của nó trong Excel 2013, các bạn sử dụng Excel 2010 hoặc 2007 sẽ có thể thấy cái tương tự , không giống 100% đâu nhé):



Với bảng công cụ này, các bạn có thể làm được mọi thứ cần thiết với VBA như viết code, thiết kế… nhưng hãy lưu ý rằng: Excel VBA chỉ có thể hoạt động khi chế độ bảo mật “Macro Security” được kích hoạt ở trạng thái enable.

Tự học VBA – Bài 2: The Excel VBA Editor – Bộ biên dịch code

Ở bài trước, chúng ta đã tìm hiểu về cách thức để mở lên “Ví Thạch Sanh” trong lập trình Excel VBA. Đại khái trông nó giống như hình dưới đây:



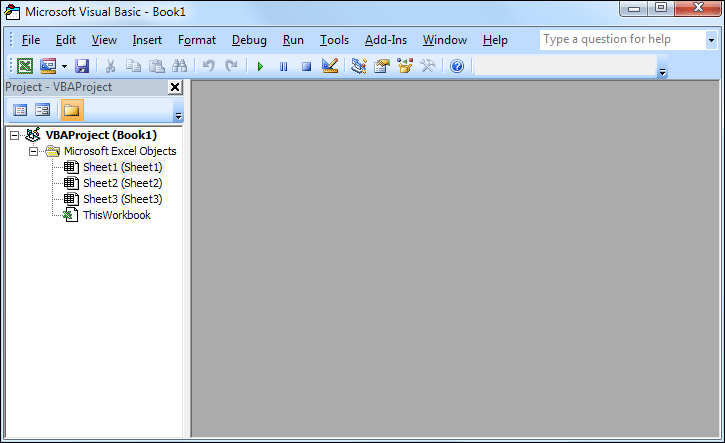
Trong phần bài học này, chúng ta sẽ tìm hiểu về nút ngoài cùng bên trái: “**Visual Basic**” xem công dụng của nó ra sao nhé. Nội dung bài học khá dài vì vậy các bạn cần có những cố gắng nhất định để chinh phục hành trình [tự học VBA](http://www.h2account.com/tu-hoc-vba-trong-excel) thành công.  Từng bước từng bước 1, Tôi tin vào sự cố gắng của các bạn và chúng ta cùng bắt đầu ngay nào:

**The Excel VBA Editor – Bộ biên dịch code**

Để lập trình chúng ta phải có chỗ để viết code, với VBA trong Excel người ta gọi nó với cụm từ: “The Excel VBA Editor” tạm dịch là “Bộ biên dịch code”. Có một số cách để mở bộ biên dịch này như sau:

*Cách 1: Từ công cụ****Developer****–> Click**nút****Visual Basic****.*  
*Cách 2: Sử dụng tổ hợp phím****Alt + F11***

Với cách nào đi nữa thì bạn cũng sẽ mở được bộ biên dịch code VBA và trong nó giống như hình dưới đây. Hãy nhớ rằng nếu bàn phím và con chuột của bạn hoàn toàn bình thường mà sau khi bạn thực hiện theo một trong hai cách trên mà không thấy hiện hình dưới này thì có nghĩa là Excel của bạn đã bị lược mất phần VBA –> Khắc phục bằng cách cài lại bản Office chuẩn theo [**LINK NÀY**](http://www.h2account.com/video-huong-dan-cai-dat-office-2010.html)



Bạn thấy thế nào, liệu có câu hỏi nào trong đầu bạn kiểu như: Excel còn chưa thông giờ lại mọc ra một cái thằng như thế này thì học làm sao được? ^\_\_^    Hãy cứ bình tĩnh vì vạn sự khởi đầu nan. Gian nan thì chớ có nản…

**Xem nào, có một vài thứ cần phải giai thích đối với bộ biên dịch VBA này:**

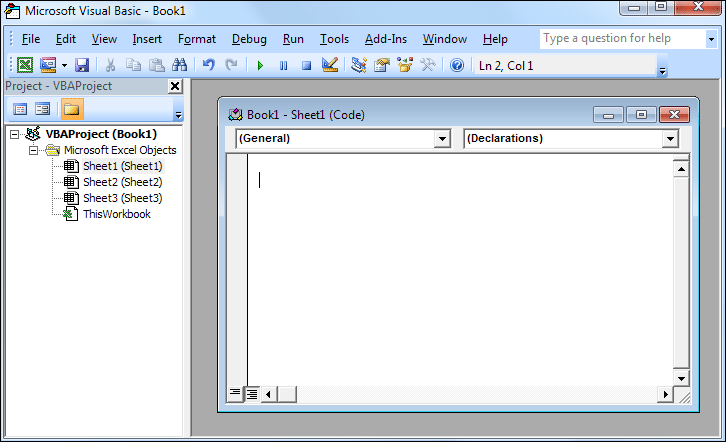
Trước tiên, khu vực màu trắng bên trái được gọi với tên: **Project – VBA Project** là nơi lưu trữ toàn bộ những dự án VBA của bạn có trong file excel đang mở. Mặc định, sẽ có một số thành phần sau: Sheet1, Sheet2, and Sheet3 (Với Excel 2013 chỉ có 1 worksheet mặc định, vì vậy bạn sẽ chỉ nhìn thấy Sheet1 khi mở bộ biên dịch code VBA trên phiên bản này). Bạn có thể thấy rằng: Sheet1, sheet2, Sheet3 là đại diện cho các bản tính hiện thời trong file excel và khi click vào bạn sẽ thấy hiện ra vùng để cho bạn viết code cho sheet đó, vậy cái thứ 4 : ThisWorkbook là gì? Nó là nơi lưu trữ những dự án VBA của bạn áp dụng cho Workbook tức là toàn bộ file excel chứ không phải đơn thuần một sheet nào cả.

*Chú ý: Hãy thử nghịch nó nếu bạn thích, tắt đi cũng được và nếu muốn bật lên thì sử dụng tổ hợp phím Ctrl + R*

Một đối tượng khác không được liệt kê trong danh sách này là các Macro – nghe có vẻ khá nguy hiểm nhỉ. Chúng ta sẽ tìm hiểu kĩ hơn về Macro trong bài 3, các bạn chỉ cần hiểu là khi có Macro trong file excel thì nó cũng sẽ liệt kê ở VBA Project

Vùng màu xám lớn là nơi bạn sẽ viết những dòng code của bạn. Nó có màu xám là vẫn chưa có cửa số viết code nào được mở. Để mở những cửa sổ này, trên màn hình hiển thị bộ biên dịch code các bạn bấm đúp vào một đối tượng tại khu vực VBA Project: Có thể là Sheet hay Workbook đều được.

Thử click đúp vào Sheet1 thì chúng ta sẽ được giống như hình dưới đây:



Tất cả các ứng dụng của bạn sẽ được đánh vào vùng trắng mới xuất hiện này. Nó là dạng cửa sổ bạn có thể ấn nó đi, phóng to, thu nhỏ hoặc tắt hẳn. Mở nó lại bằng cách nhấp đúp vào dự án bên trái tương ứng. Có thể là Sheet1…, là ThisWorkbook, Macro hoặc một tên nào đó bạn đặt cho dự án của mình. Trên cửa sổ mới xuất hiện có hai vùng nhỏ phía trên tạm thời chúng ta cứ để đó và tìm hiểu nó vào các bài kế tiếp nhé.

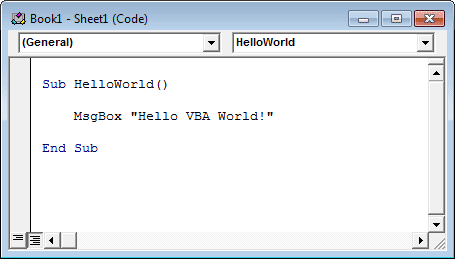
Trở lại với bài học khi các bạn đã mở cho mình vùng nhập code bằng cách click đúp vào Sheet1 rồi. Hãy thử viết một chương trình VBA đầu tiên rất đơn giản như sau:

**Sub HelloWorld()**

**MsgBox “Hello VBA World!”**

**End Sub**

Hì hục viết code xong thì nó trông giống như thế này đây:



Chúng ta sẽ giải thích cặn kẽ từng phần một của đoạn mã code này để các bạn có thể hiểu rõ các thành phần của một chương trình trong VBA

*Cấu trúc:*

**Sub ( hoặc Private Sub) + tên chương trình + ()**

***Đoạn mã code***

**End Sub**

Từ khóa **Sub** là từ viết tắt của từ Subroutine (Chương trình con). Nó là phần đầu tiên và bắt buộc phải có trong mọi dự án của bạn. (Ngoài Sub, VBA còn có thể bắt đầ bằng Private Sub).  Để kết thúc một chương trình chúng ta dùng End Sub. Khi gõ “Sub tenchuongtrinh” sau đó ấn enter thì lập tức End Sub sẽ được tạo luôn cho người viết.

Một Sub cần một tên đi cùng theo sau bởi một cặp ngoặc đơn()  (<– Đó là quy định trong VBA và chúng ta phải chịu khó tuân theo thôi). Các bạn cũng lưu ý là có một khoảng trắng giữa Sub và tên. Ở giữa Sub và End Sub là nơi bạn viết các dòng code cho chương trình của bạn. Bạn có thể đặt mọi thứ bạn thích nhưng phải đảm bảo sự tương thích và hợp lý để chương trình có thể hoạt động được.

Lưu ý: Tên chương trình của bạn không được chứa khoảng cách, dấu tiếng Việt

***Sub Hello\_World()   <— ĐÚNG***

***Sub Hello World()   <—- LỖI***

***Hãy lưu ý những điểm sau trong khi đặt tên chương trình của bạn:***

- Không được đặt tên sub bắt đầu bằng các con số: 0,1,2,3…  
- Không được dùng các dấu ngắt nghỉ (dấm chấm, phẩy…) trong tên của Sub  
- Bạn không được phép sử dụng các ký tự đặc biệt: #, $, %, &, !

Trở lại với chương trình đầu tay của bạn:

**Sub HelloWorld()**

**MsgBox “Hello VBA World!”**

**End Sub**

Các bạn đã biết về Sub, End Sub, HelloWorld() vậy: Đoạn ở giữa: MsgBox “Hello VBA World!”  là gì ?

Đó là phần code của chương trình HelloWorld trong đó:

**MsgBox** là hàm xuất nội dung ra ngoài màn hình (Các bạn có thể thấy phần mềm kế toán excel H2 Account có rất nhiều thông báo ra ngoài màn hình khi các bạn thao tác là ứng dụng cái này đó)

“Hello VBA World!” có chứa một cái ngoặc kép “” và nội dung bên trong. Nội dung bên trong thì các bạn có thể tùy ý thay đổi còn cái ngoặc kép thì nó là quy định của hàm rồi. chúng ta phải làm đúng để VBA có thể hiểu được ý đồ của người viết.

**Vậy có bao nhiêu hàm trong VBA ??? Câu trả lời là có rất nhiều**

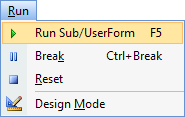
**Làm thế nào để có thể ghi nhớ được chúng??? Giống như học ngoại ngữ , muốn ghi nhớ chúng ta phải thực hành thường xuyên.**

**Ghi nhớ các hàm này có lâu và khó không? Câu trả lời là đơn giản thôi —> PHÙ, không khó rồi cố gắng lên nhé các bạn**

Khi một chương trình đã hoàn thành thì chúng ta cần phải chạy nó để kiểm tra xem chương trình có gặp lỗi nào không. Để chạy thì các bạn để ý lên thanh công cụ phía trên có nút tam giác màu xanh, các bạn click vào nút này để chạy

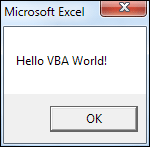
VBA Tools - 01

Một chác khác để chạy thử các dòng code là click vào Menu Run –> Click Run Sub/UserForm



Còn muốn nhanh hơn nữa thì các bạn bấm phím tắt: F5 trên bàn phím

Với chương trình trên thì các bạn sẽ nhìn thấy một hộp thoại thông báo sau khi chạy như sau:



May mắn là chúng ta không gặp lỗi nào nghiêm trọng để chương trình không chạy. Các bạn thấy bảng này thì có thể tự tin rằng mình đã là một coder nghiệp dư rồi đó. Mọi việc đều phải bắt đầu từ những thứ đơn giản, sau những lần thực hành, trải nghiệm, suy nghĩ trình độ các bạn sẽ nâng dần theo thời gian.

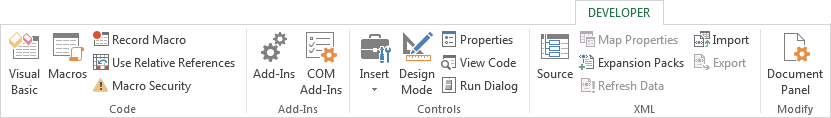
Kết thúc các bài học trên website này, tôi tin rằng các bạn sẽ trở thành những người có kiến thức về VBA và nếu bạn chịu khó tìm hiểu và học hỏi thêm thì rất có thể bạn sẽ xây dựng được chương trình hoành tráng như [phần mềm kế toán excel](http://www.h2account.com/)H2 Account chẳng hạn. Biết đâu…

Tự học VBA – Bài 3: Tìm hiểu cách ghi Macro trong Excel

Có phải bạn đang nóng lòng muốn bắt tay vào các bài tập thực hành để sớm tự xây dựng được những chương trình của riêng mình? Mình cũng đang rất giống các bạn chỉ có điều mình nóng lòng muốn các bạn hãy suy nghĩ cẩn thận và thấu đáo từ Bài 1, Bài 2 và cho đến bài 3: Tìm hiểu cách ghi Macro trong Excel.

**Nếu như:**

**Bài 1:** Chúng ta biết cách để mở thanh công cụ dành cho lập trình: Developer  
**Bài 2:** Chúng ta làm quen với bộ biên dịch code, và cũng là tìm hiểu về cái nút đầu tiên trong bộ tính nắng rất hấp dẫn mà các bạn nếu chưa nhớ kĩ thì có thể nhìn lại nó dưới đây:



(Áp dụng cho Excel 2013, các phiên bản Excel 2010, 2007 có thể tương tự)

Sự thật là, ngay bên cạnh nút “Visual Basic” là một bộ liên quan đến Macros, sẽ có nhiều câu hỏi xung quanh vấn đề này cần câu trả lới thấu đáo và qua đó, những lập trình viên VBA tương lai có thể tận dụng được sức mạnh của Macros để có thể tự học hỏi và nâng cao khả năng viết code của mình. Chúng ta cùng bắt đâu nhé…!

**Macro là gì ?**

Macro là một tập hợp các lệnh được phần mềm ghi lại các thao tác trên phần mềm đó theo một kịch bản dựng sẵn và các dòng lệnh này có thể được khởi chạy lại để làm các công việc khác giống như đã được ghi trước đó một cách tự động.

**Macros có thể ứng dụng như thế nào ?**

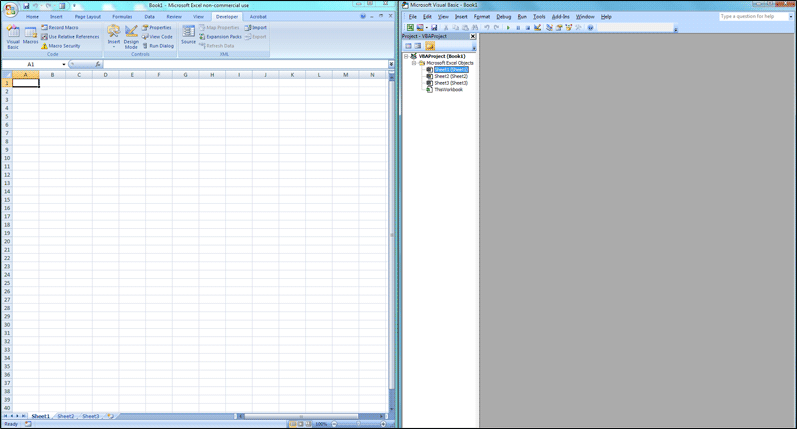
Chính bởi khả năng có thể làm lại giống hệt các thao tác chúng ta đã ghi, nên macros được ứng dụng để giảm thiểu các thao tác lặp đi lặp lại cho người dùng, mục đích chính vẫn là để tiết kiệm thời gian.

**Làm thế nào để tạo ra được Macro?**

Có hai cách phổ biến nhất là: **Sử dụng tính năng ghi Macro của phần mềm** hoặc**viết code.**Trong phần tiếp theo chúng ta sẽ tìm hiểu cách ghi Macro trong excel và các bài học tiếp theo nữa chúng ta sẽ đi sâu vào phần viết code.  Lý do chúng ta bắt buộc phải tìm hiểu quá trình ghi macro trong excel là bởi thông qua việc tìm hiểu này, các bạn sẽ có cái nhìn trực quan nhất về việc hình thành trong đầu những bước viết ra những dòng code theo trình tự khoa học, ngoài ra thì việc xem excel tự viết code cũng là một cách học hỏi rất thú vị đối với những người mới bắt đầu. Đả thông tư tưởng rồi thì cố gắng tiếp nhé các bạn! COME ON…!

Chú ý: khả năng ghi Macro trong Excel là có giới hạn, bạn sẽ hơi ngợp bởi lượng code mà excel sinh ra nhưng thực tế là với kiến thức bạn được trang bị ở các bài kế tiếp, các dòng code thực tế để chạy được chương trình excel ghi sẽ ngắn hơn rất nhiều.

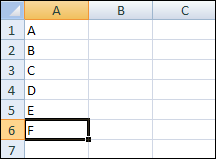
Trước tiên, hãy chắc chắn rằng bạn đang mở excel và trình biên dịch code (Alt + F11) và đặt chúng ở cạnh nhau tương tự như hình dưới đây.



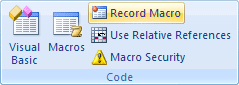
Nếu bạn đang xài windows 7 hoặc mới hơn, bạn có thể đơn giản có hai chương trình mở dạng hai cửa sổ liền nhau bằng cách làm theo hướng dẫn này:

Bật Excel và Alt + F11 –> Click vào tab excel trên thanh công cụ start ngoài màn hình desktop –> Bấm giữ phím windows + Nút sang trái –> Excel sẽ được thu gọn lại một nửa màn hình bên trái. Chuyển qua tab bộ biên dịch code –> Bấm giữ phím windows + Nút sang phải –> Toàn bộ bộ biên dịch code được co lại một nửa màn hình bên phải. Như vậy xem rất ok rồi.

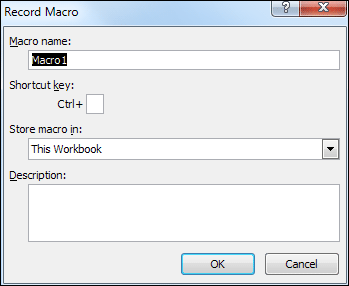
Và giờ thì chúng ta sẽ chuẩn bị một số thứ đơn giản đầu tiên để theo dõi quá trình ghi Macro của excel . Đầu tiên, hãy đánh các ký tự ở cột A dòng từ 1 đến 6 như hình dưới đây: (Nhớ để yên đừng có căn chỉnh gì đấy nhé)



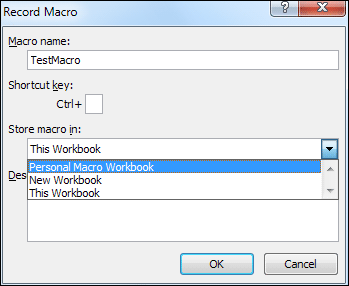
Trên thanh công cụ Developer, các bạn bấm vào biểu tượng như hình bên dưới (Lưu ý: khi bạn co màn hình lại thì Excel sẽ không hiển thị đủ tên: Record Macro, các bạn nhìn biểu tượng và bấm thôi chứ đừng tìm chữ nhé)



Ngay khi bạn bấm nút ghi thì một hộp thoại sẽ hiện lên hỏi tên của macro là gì, phím tắt, nơi lưu trữ macro và mô tả cho macro bạn định ghi:

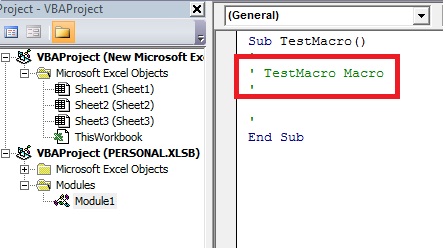


Đối với tên, các bạn có thể đặt tên tùy ý làm sao mà có thể phân biệt được là ok, còn trong phạm vi bài này mình tạm đặt: TestMacro. Click nút sổ xuống tại **Store macro in** và chọn **Personal Macro Workbook:**(Nếu bạn thắc mắc chọn mấy cái bên dưới nó ra cái gì thì sao không tự mình thử khám phá và so sánh nhỉ ^\_^)

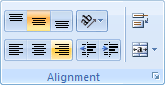


Đặt xong mấy cái này thì các bạn bấm OK. Ngay sau khi bấm OK các bạn quan sát bộ biên dịch code – phần bên trái lưu trữ các dự án của mình có gì đặc biệt không ? –> À, thì ra là sự xuất hiện của một cây thư mục VBAProject(Personal.XLSB) –> Các bạn bấm tiếp vào dấu **+** ở ngoài thư mục này –> Bấm tiếp **Modules** –> Click đúp vào**Module1** –> Quan sát bên phải

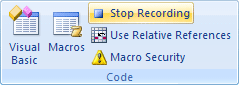
Một cặp Sub và End Sub đã được tạo ra với tên là TestMacro vậy còn cái dòng màu xanh lá cây kia có nghĩa là gì? Đó là ghi chú trong VBA . Viết ghi chú trong VBA bằng cách sử dụng dấu nháy đơn (dấu này đặt cạnh nút enter ý). Excel nó sợ chúng ta khi nhìn vào tên của macro vẫn chưa thể hình dùng ra cái gì nên phải ghi chú rất cụ thể. Đối với một số chỗ quan trọng dễ gây nhầm lần trong [**phần mềm kế toán excel**](http://www.h2account.com/)**H2 Account**, đội ngũ phát triển cũng chèn ghi chú để tiện kiểm soát đó.

[](http://www.h2account.com/wp-content/uploads/2015/05/TestMacro-01.jpg)

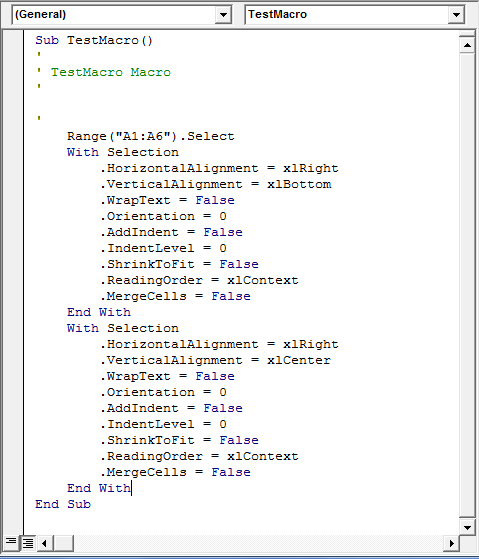
OK, chúng ta tiếp tục ghi macro nào, nãy giờ các bạn đừng click chuột lên bàn phím hoặc định dạng bất cứ cái gì nhé. Các bạn chọn các cell từ A1 tới A6, Click menu HOME trên thanh công cụ phía trên –> Chọn canh lề phải và căn đều trên dưới như hình dưới đây: (Vừa chọn vừa quan sát sang bộ biên dịch VBA để xem nó hoạt động như thế nào nhé)



Chuyển nhanh qua tab Developer. Các bạn bấm vào biểu tượng Stop Recording – Hình vuông xanh để dừng việc ghi



Bây giờ chúng ta sẽ quan sát cửa sổ code để xem thành quả của việc ghi vừa rồi:

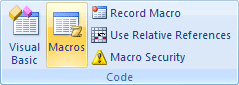
[](http://www.h2account.com/wp-content/uploads/2015/05/TestMacro-02.png)

Hơi căng thẳng nhỉ …Bình tĩnh chúng ta chưa cần phải hiểu hết cái mớ hỗn độn này làm gì cả. Việc của các bạn để bộ não thư giãn, trong các phần tiếp theo chúng ta sẽ giải thích cặn kẽ trong cái mớ hỗn độn ấy nó thường gồm những gì? Thao tác ra làm sao, tuần tự như thế nào…

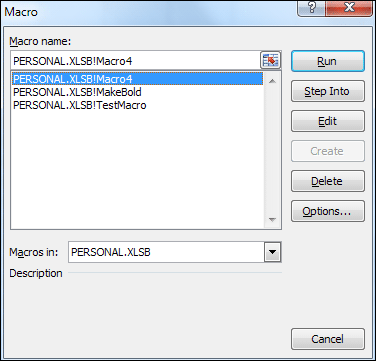
**BÀI TẬP THỰC HÀNH**  
Ghi các macro mới với việc định dạng lại các chữ cái từ cell A1 đến A6 với:  font chữ khác, cỡ chữ khác, và màu chữ khác

Khi bạn hoàn thành các bài tập trên sẽ thấy một điểm chung là excel sinh ra rất nhiều mã cho những thao tác rất đơn giản. Thực tế để làm được chúng thì chỉ cần vài dòng code đơn giản mà thôi.

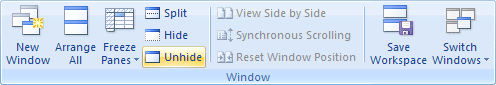
Trước khi các bạn tiếp tục với các bài khác, cách tốt nhất là xóa hết các macros đã làm trước đó bằng cách click vào tab Developer trên thành công cụ –> Click Macros



Bạn sẽ nhìn thấy một hộp thoại liệt kê danh sách các macros như hình dưới, các bạn chọn lần lượt các macro cần xóa và bấm nút Delete bên phải.



Nếu bạn nhận được một thống báo lỗi rằng excel không thể xóa được các macros trong khi một sheet nào đó đang bị ẩn –> Các bạn click Cancel tại hộp thoại thông báo và khắc phục bằng cách sau: Trên thanh công cụ –> Click chọn **View** –> Chọn **Unhide**



Trong bài này, các bạn đã có thể hiểu được phần nào cách thức hoạt động của việc ghi macro đơn giản. Các bạn có thể áp dụng thử nâng cao với nhiều trường hợp khác nhau, phức tạp hơn theo cách nghĩ của bạn. Đừng quên quan sát sự thay đổi sau mỗi lần chọn lựa. Có thể bạn không hiểu những dòng sinh ra ngay nhưng không sao hết. Mới chỉ có bài 3 và rất nhiều điều thú vị khác đang chờ bạn khám phá ở các bài tiếp theo. Cố gắng lên!

Tự học VBA – Bài 4: Ký hiệu dấu chấm trong VBA

Sau 3 bài mở đầu…các bạn thấy quá trình [tự học VBA](http://www.h2account.com/tu-hoc-vba-trong-excel) thế nào rồi? Các bạn đã thử nghiên cứu lại về Record Macro để tìm hiểu thêm một số cách thức mà excel sinh ra những dòng code để thực hiện hóa các công việc một cách tự động chưa?

Trong bài 4: Các bạn sẽ tìm hiểu về một đặc trưng trong việc viết code VBA đó là việc sử dụng dấu chấm (.) , và chỉ cần sai một dấu nhỏ thôi là phần mềm sẽ báo lỗi.

Excel VBA sử dụng ký hiệu chấm (.) để tách những phần khác nhau mà bạn có thể truy cập và điều khiển bằng ngôn ngữ lập trình. Ký hiệu chấm thể hiện sự phân cấp và thông thường chúng ta bắt đầu với một đối tượng (**object**) (Trong Excel, một đối tượng là thứ bạn có thể thao tác chẳng hạn như worksheet). Sau đối tượng, bạn gõ một dấu chấm, rồi tiếp tục xác định điều bạn muốn làm hoặc điều khiển với đối tượng này. Người ta gọi đó là PHƯƠNG THỨC (**method**). Quá trình điều khiển được thực hiện thông qua THUỘC TÍNH (**properties**) hoặc THÔNG SỐ (**parameters**)

Bạn vẫn thấy khó hiểu, chúng ta hãy thử tưởng tượng đơn giản như thế này:

Bạn nghĩ về một cái TIVI –> Nó là một đối tượng và chúng ta có thể ghi chú nó tương tự như thế này:

**tv**

OK, mọi thứ đều đang có vẻ rất đơn giản, chỉ là một cái tv. Nhưng, bạn sẽ cần một vài thông tin nếu bạn quyết định đi mua một cái tivi. Một trong số đó là cái tivi của bạn sẽ to như thế nào?  Để chèn thuộc tính kích cỡ (size) bạn sẽ làm như thế này:

**tv.size**

Và bạn muốn kích cỡ của nó rõ ràng hơn là 55 inch bạn sẽ làm như thế này:

**tv.size = “55 inch”**

Chúng ta biết có một đối tượng là cái TV (tv) và một thuộc tính là kích cỡ (size). Và chúng ta cũng có giá trị cho kích cỡ (size) là 55 inch.

Nếu chúng ta muốn sở hữu cái tivi này trước tiên chúng ta phải thực hiện một một hành động là MUA (buy). Chúng ta có thể gọi hành động MUA (buy) là một phương thức của cái tv:

**tv.buy**

Phương thức có thể đi cùng với các thiết lập thêm, người ta gọi nó là thông số. Một thông số của phương thức mua là hình thức thanh toán (PaymentType). Lẽ dĩ nhiên hình thức thanh toán (PaymentType) đi liền với một số giá trị cụ thể như: Tiền mặt, Tiền gửi, Séc, Vocher…) Chúng ta có thể viết như bên dưới để mô tả cho trường hợp phương thức có thông số đi kèm:

**tv.buy PaymentType:=TienMat**

Chúng ta có một khoảng trống giữa phương thức (mua) và thông số (Hình thức thanh toán – PaymenType). ***Giá trị của thông số*** được viết sau dấu hai chấm bằng (:=) và ***không có khoảng trống ở giữa.***

Chúng ta có thể có nhiều thông số cho phương thức mua, chẳng hạn như chúng ta có thể có thông số: Giảm giá (Discount), Quà tặng đi kèm(gift).

**tv.buy PaymentType:=TienMat Discount:=No Gift:=No**

**Một số Lưu ý:**

*– Chúng ta sử dụng một dấu cách để phân tách các thông số cùng giá trị của chúng.*

*– Để phân biệt nhanh Phương thức và Thuộc tính có thể hình dung: Phương thức là ĐỘNG TỪ, thuộc tính là TÍNH TỪ (Động từ như: Mua, bán, chọn, đập vỡ, thay thế…, Tính từ như: Kích cỡ, màu sắc, trọng lượng, chiều cao…) Nhớ là hình tượng thôi còn khi áp dụng vào excel VBA sẽ có các thuộc tính và phương thức rõ ràng cho từng đối tượng mà các bạn phải tuân theo.*

**Thuộc tính trong VBA**

Nào, giờ thì tạm chia tay với ước mơ TIVI 55 inch để trở về với excel. Khi lập trình VBA, các bạn sẽ sử dụng rất nhiều: Đối tượng, phương thức, thuộc tính và thông số. Mình đi vào ví dụ cụ thể luôn. Chúng ta có một đối tượng gọi là ActiveCell (Đây là ô mà con chuột của bạn đang hiện thời ở đó). Đối tượng ActiveCell có thể có một thiết lập font chữ cho nó:

Theo các bạn: font chữ là thuộc tính hay phương thức? –> Dĩ nhiên là thuộc tính rồi bởi chúng ta không thấy có hành động nào ở đây đối với ActiveCell cả.

**ActiveCell.Font**

Tuy nhiên, Font lại là tên của một thuộc tính cha mà nó bao hàm rất nhiều các thuộc tính con khác: như Tên font, Kiểu đậm, nghiêng… Vì vậy, sau Font chúng ta tiếp tục sử dụng một dấu chấm khác viết Name (tên) liền sau dấu chấm đó.

**ActiveCell.Font.Name**

Bởi vì Font.Name là một thuộc tính, bạn sẽ cần một giá trị cụ thể cho nó, hãy thử với một kiểu font rất quen thuộc trong bộ gõ Unicode:

**ActiveCell.Font.Name = “Times New Roman”**

Như vậy font chữ “Time New Roman” là một giá trị được áp dụng cho đối tượng ActiveCell.

Đến đây, hãy bình tĩnh phân biết giữa GIÁ TRỊ và THÔNG SỐ nhé các bạn

Chúng ta cũng có thể thiết lập định dạng chữ đậm cho thuộc tính Font bằng cách sau:

**ActiveCell.Font.Bold = True**

Lại phải bình tĩnh và không được đánh đồng cho rằng giá trị phải rõ ràng như là các con số, cụm từ cụ thể. Ở đây chúng ta có đối tượng ActiveCell. Theo sau nó là thuộc tính Font và ở trong Font lại có thuộc tính con là Bold (Đậm). Với Bold chúng ta chỉ có hai hình thái hoặc là đậm hoặc là không đậm. Hai hình thai này ứng với giá trị True (Đúng) và False (Sai). Đúng thì là đậm mà sai thì không. Bạn phải tưởng tượng tiếp các thuộc tính con tương tự như: in nghiêng, gạch chân cũng chỉ có giá trị True or False, nhưng nếu thuộc tính con là cỡ chữ (size) thì giá trị là phải bằng một con số rõ ràng.

**Vậy còn phương thức trong Excel VBA thì sao?**

Như đã nói với các bạn từ ví dụ cái tivi, khi lập trình VBA chúng ta phải tuân theo những phương thức được quy định sẵn ứng với từng đối tượng. **TRỜI ƠI, VẬY LÀM SAO MÀ BIẾT ĐƯỢC CÁC PHƯƠNG THỨC VÀ THUỘC TÍNH CỦA CÁC ĐỐI TƯỢNG CHỨ** –> Đừng lo, vì VBA có gợi ý cho bạn, chỉ là bạn phải chịu khó học thêm vốn từ tiếng Anh nho nhỏ để có thể nhìn mặt bắt hình dong hoặc chịu khó sử dụng kỹ thuật tìm kiếm trong VBA để biết mình phải làm cái gì. Chúng ta sẽ trao đổi cụ thể hơn ở các bài tiếp theo. Xin cứ bình tĩnh và tiếp tục với một ví dụ về phương thức Quit (Đóng)

**Application.Quit**

**OMG – Định troll mình hay sao mà lúc thì ActiveCell lúc thì Application vậy trời?**Rốt cục thì có những đối tượng nào để tôi còn biết mà lập trình chớ…@@  –> Trả lời: Có đến hơn 100 đối tượng để các bạn có thể lôi ra làm thí nghiệm cụ thể sẽ được trình bày ở các bài học tới, xin chớ nóng vội mà hỏng việc. Mặc dù có nhiều thế nhưng chúng ta chỉ hay làm việc với một số đối tượng nhất định thôi, làm nhiều sẽ nhớ nên các bạn cứ yên tâm.

Quit tức là đóng, mà đóng lại là động từ rồi, động từ này tác động trực tiếp nên đối tượng Application nên Quit là một phức thức chuẩn của chỉnh.

**LƯU Ý: TRANH THỦ VỪA ĐỌC VỪA HỌC LUÔN MỘT SỐ TỪ TIẾNG ANH CĂN BẢN SẼ CÓ ÍCH VỀ SAU VÀ ĐẶC BIỆT HỮU DỤNG KHI CÁC BẠN ĐỌC TÀI LIỆU NƯỚC NGOÀI.**

Một ví dụ khác về phương thức là:  Add (thêm, chèn).Bạn có thể sử dụng nó để thêm một worksheet mới vào file Excel của bạn (Worksheet là cách nói hoa mỹ thôi, chứ nó chỉ là sheet ấy mà ^^)

**Worksheets.Add After:=Worksheets(1)**

Đối tượng trong ví dụ trên là Worksheets. Và Add là phương thức được điền ngay sau dấu chấm. Quan sát tiếp bạn sẽ thấy một khoảng cách rồi đến After. ??? Vậy Affter là gì nhỉ ??? Thử nhớ lại xem…À đúng rồi, nó là thông số. Thông số After cho ta biết sẽ có một sheet được chèn vào sau một worksheet . Trong ví dụ là Sheet1. Tại sao lại là sheet1? Bởi vì giá trị worksheet(1) cho ta biết điều đó. ( bên trong dấu ngoặc đơn, bạn có thể nhập tên của sheet hoặc số thứ tự của sheet đó )

**Vùng dữ liệu (The Range Property)**

Phần lớn các macros bạn viết sẽ cần tới tham chiếu vùng dữ liệu (Range). Nhưng vùng dữ liệu là gì?

Một vùng dữ liệu đơn giản là một vùng tập hợp các ô (cells) trên một bảng tính(sheet). Một vùng dữ liệu có thể là một ô, một tập hợp các ô. Vùng dữ liệu bạn muốn tham chiếu được đặt trong một cặp ngoặc đơn như thế này: () –> Range(“…”). Vậy câu hỏi được đặt ra là: trong dấu ngoặc đó có những trường hợp tham chiếu như thế nào?

***Chúng ta có thể liệt kê như sau:***

**Range(“A1″)** – Đây là một ví dụ chỉ tham chiếu tới ô A1

**Range(“A1:B7″)** - Cú pháp này chúng ta sẽ tham chiếu tới vùng A1:B7

Chú ý: Phải có dấu ngoặc kép bao bọc vùng tham chiếu, có dấu hai chấm ở giữa hai cell A1 và B7 và không có khoảng các nào giữa hai cell này nhé các bạn. Để tránh nhầm lẫn chúng ta nên quy wocs : Tham chiếu đầu tiên phải ở bên phải, phía trên. Tham chiếu sau phải ở bên trái, phía dưới hoặc ngang bằng.

**Range(“A1″, “B7″)** – Một cách khác để tham chiếu vùng dữ liệu từ A1 tới B7.

–> Chúng ta có rất nhiều kiểu tham chiếu khác nhau, tham chiếu cột, tham chiếu dòng, tham chiếu nhiều vùng một lúc…các bạn khi cần tham chiếu đến đâu thì Google đến đó nhé. Hoặc chịu khó tư duy từ đó sẽ nhớ lâu hơn là mình show hết ra để rồi bị ngộp là thấy nản ngay.

Một khi bạn đã có vùng dữ liệu thì nó được gọi là một đối tượng và chúng ta cần phải có hành động đối với đối thương này. Hay chính là tìm một phương thức cho cái vùng dữ liệu đó. Một phương thức đó là select (lựa chọn) – nó tương đương với việc bạn đặt con chuột vào vị trí ô A1. Khi lập trình VBA tức là chúng ta đã bỏ qua khái niệm dùng chuột mà mọi thứ đều phải được tính toán hết trong đầu người lập trình rồi. Vì thế các bạn cần phân biệt rõ Range(“A1″) và Range(“A1″).select nhé.

**Range(“A1″).select**

**Range(“A1:B7″).select**

**Đối tượng Worksheets – The Worksheets Object**

Chúng ta vừa đề cập ở trên về vùng dữ liệu, nhưng vùng dữ liệu này thuộc cái gì? Đó chính là đối tượng worksheets. Thông thường khi lập trình, để cụ thể chúng ta sẽ sử dụng đối tượng worksheets đầu tiên, kế đến là vùng dữ liệu. Khi đó nó giống như thế này:

**Worksheets(“Sheet1″).Range(“A1″).Select**

Bạn có thể thấy, đầu tiên, chúng ta truy cập đến worksheet có tên là Sheet1. Ngoài tên thì bạn cũng có thể điền số ứng với vị trí tương ứng của sheet đó. Nếu các bạn để ý, thì vùng lưu trữ dựa án VBA của bạn thì ở phần Microsoft Excel Object – Chỗ mấy cái sheets nó sẽ gồm hai phần: Phần tên sheet theo người dùng đặt và phần tên Sheet theo số thứ tự của hệ thống – Bạn có thể nhờ đó để biết chính xác sheet của mình có số là bao nhiêu. Khi đã rõ ràng, bạn có thể viết tương tự như dưới đây:

**Worksheets(1).Range(“A1″).Select**

Nãy giờ chúng ta nói nhiều về Sheet1. Tuy nhiên, bạn hoàn toàn có thể tham chiếu tới mọi sheet hiện có trong worksheet s của mình như sau.

**Worksheets(2).Range(“A1″).Select**

hoặc

**Worksheets(“Sheet2″).Range(“A1″).Select**

**Các lưu ý:**

1)  Khi bạn đang ở Sheet2 (hoặc một sheet bất kì không phải là sheet1) Mà trong câu lệnh VBA trong một cặp sub các bạn gõ: Worksheets(1).Range(“A1″).Select sẽ thấy bị báo lỗi 1004 – Đối tượng không thể tìm thấy. Như vậy có thể suy ra rằng: Trong VBA tính hiện thời là rất quan trọng. Việc bạn ở một nơi nhưng lại muốn tham chiếu đến một nơi khác cần có sự thao tác tuần tự để tránh lỗi phát sinh. Chúng ta có thể viết:

*Sub*  
*Worksheets(1).Select*  
*Range(“A1″).Select*  
*End sub*

2) Cần phân biệt rõ giá trị ở trong mỗi dấu ngoặc đơn giữa Worksheets và Range . Nếu như trong Worksheets các bạn sẽ thấy giá trị nằm trong ngoặc đơn bình thường : VD: (giatri) thì ở trong Range, giá trị nằm trong ngoặc kép và tất cả lại nằm trong ngoặc đơn. VD: (“giatri”)  –> Nếu các bạn không chú ý đến điều này khi code rất dễ phát sinh lỗi mà không biết mình sai ở đâu.

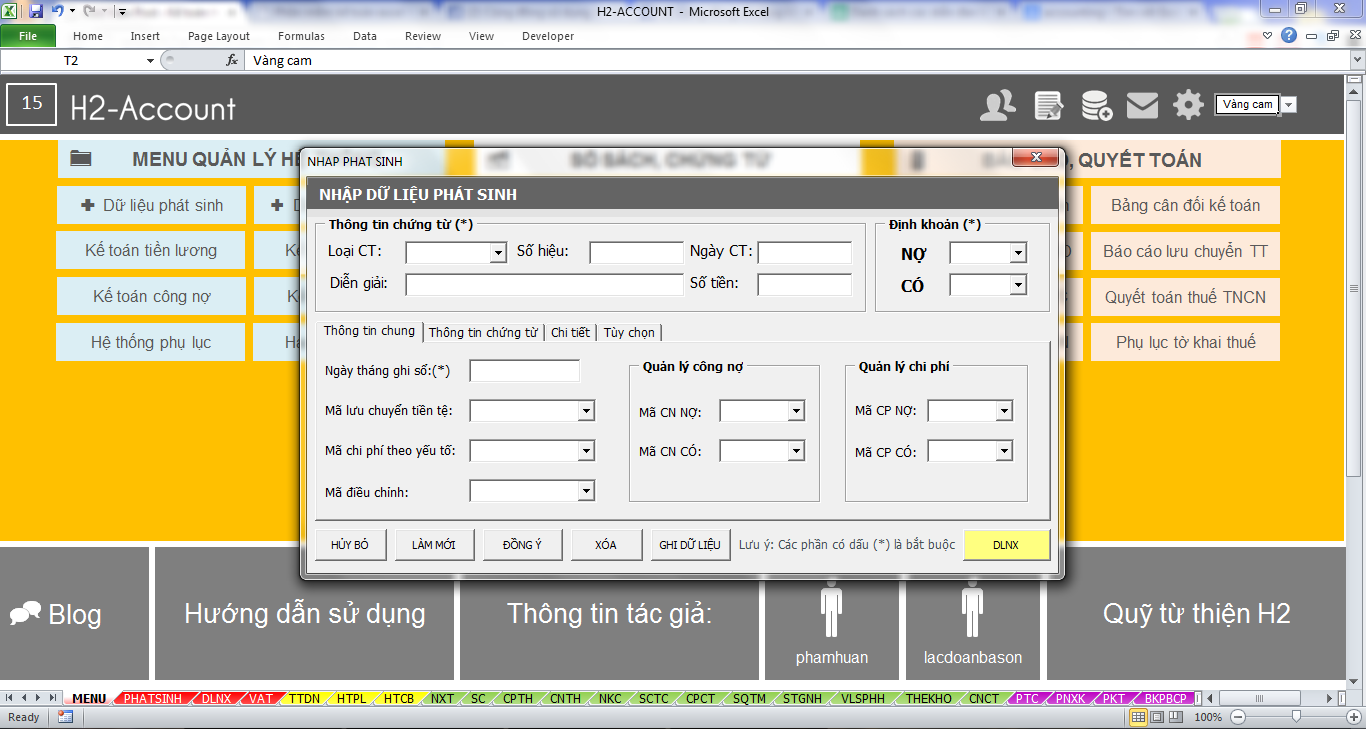
Bài này có nội dung khá dài và nhiều thuật ngữ được đề cập đến, chuyên trang về [***phần mềm kế toán excel***](http://www.h2account.com/) H2 Account muốn nhắc bạn lưu tâm đến các vấn đề sau, và để có thể trở thành một lập trình viên VBA vững chắc bạn không thể không hiểu rõ cặn kẽ:

1) Đối tượng         (***Object***)  
2) Phương thức   (***Method***)  
3) Thuộc tính       (***Properties***)  
4) Thông số          (***Parameters***)  
5) Giá trị                 (***Value***)  
6) Range                (***Vùng dữ liệu***)  
7) Worksheets     (***Sheet***)

Mọi sự thành công đều phải nỗ lực không ngừng, cánh cửa của sự hiểu biết, tinh thông VBA đang ở trước mắt bạn, có thể bạn sẽ phải đọc đi đọc lại nhiều lần bài này để hiểu cặn kẽ gốc rễ của thuật ngữ trên. Nhưng, mình tin rằng công sức của các bạn sẽ được đền đáp xứng đáng – vào một ngày không xa!

Tự học VBA – Bài 5: Tạo nút trong bảng tính

Phần lớn, khát khao của những người [**tự học VBA**](http://www.h2account.com/tu-hoc-vba-trong-excel) là muốn tạo ra một cái gì đó có quy mô và đồ sộ để phục vụ cho nhu cầu công việc của mình, một số bạn lại muốn tạo một[***phần mềm kế toán***](http://www.h2account.com/) giống như H2 Account như hình dưới đây.

[](http://www.h2account.com/wp-content/uploads/2015/01/h2vip.png)

Phần mềm kế toán excel H2 Account có giao diện rất đẹp

Cá nhân mình nghĩ rằng, không gì là không thể, chỉ là các bạn có dám ước mơ và quyết tâm dành thời gian cho nó không mà thôi. Học lập trình cũng khá giống học ngoại ngữ, chúng ta cũng phải học thuộc các từ khóa, các quy tắc lập trình cũng giống với ngữ pháp tiếng Anh. Thật khó nhớ và thật khó để  yêu thích nó được.

Kinh nghiệm của tôi nhận thấy rằng: để có thể làm được một thứ lớn, hãy thử bắt tay thực hành những thành phần nhỏ hơn trước, thuần thục, nắm rõ rồi từ từ phát triển lên. Bởi vậy, trong phạm vi của bài này, chúng ta sẽ tìm hiểu cách thức **làm thế nào để tạo ra được một cái nút có tương tác trong VBA?**

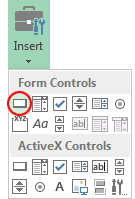
[Ở bài học trước](http://www.h2account.com/tu-hoc-vba-bai-4-ky-hieu-dau-cham-trong-vba), chúng ta đã biết cách tạo một Sub đơn giản trong cửa sổ nhập code. Trong bài học này, các bạn sẽ kích hoạt cái Sub đó thông qua một cái nút xuất hiện trong bảng tính. Hãy nhỡ rằng, mọi thứ đều được bắt đầu từ những điều đơn giản nhất.  Trước tiên, các bạn giúp mình viết một sub trong Sheet1 có sử dụng phương thức select đơn giản như sau:

*Sub AddButton ()*  
*Worksheets(1).Select*  
*Range(“A1:D6″).Select*  
*End sub*

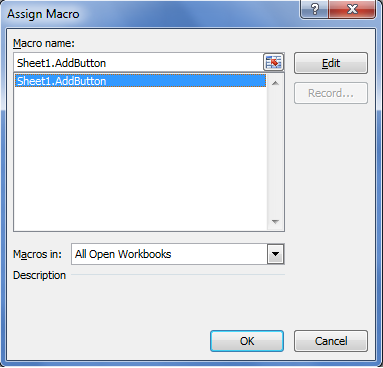
Tại phía trên của thanh công cụ VBA trong trình biên dịch code, các bạn click vào biểu tượng Excel như hình dưới đây:

Tools VBA 

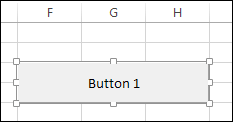
Trên thanh công cụ Developer –> Click Insert –> Click vào biểu tượng đầu tiên của form được sổ xuống. Như hình dưới đây: (Các bạn sẽ được tìm hiểu kĩ hơn về ý nghĩa của các nút trong phần học chuyên sâu về form trong VBA).



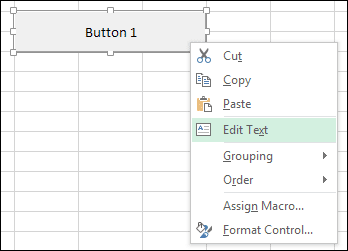
Bây giờ,  bạn di chuyển con chuột nên vùng hoạt động của sheet. Ấn giữ chuột trái của bạn vào một ô bất kì (ví dụ như ô F3 chẳng hạn). Giữ chuột và kéo xuống để vẽ một hình cái nút hình chữ nhật đến một cell khác (chẳng hạn ô H4). Ngay khi bạn thả chuột trái ra thì một cửa số sẽ hiện lên như hình dưới đây:

[](http://www.h2account.com/wp-content/uploads/2015/05/AddButtonH2-01.png)

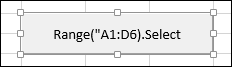
Các bạn lựa chọn Macro Sheet1.Addbuton trong list dánh sách và click OK. Excel sẽ đưa bạn đến cửa sổ hoạt động của Sheet và cái nút chúng ta vừa tạo trông nó giống như thế này:



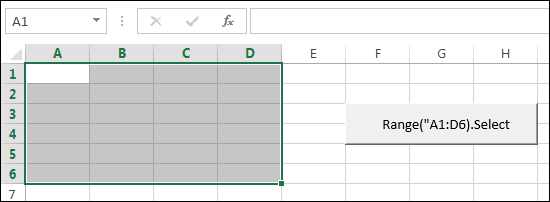
Bạn có thể chỉnh sửa nội dung trên nút một cách dễ dàng và nhanh chóng như sau: Chuột phải –> chọn Edit Text



Khi bạn lựa chọn Edit Text, một  dấu nháy chuột sẽ xuất hiện ở phần đầu text trong nút bạn vừa tạo. Đến đây thì bạn thao tác xóa chữ và nhập tên của nút theo ý muốn là được. Trong ví dụ bên dưới, để cho phù hợp với bài học và để dễ nhớ mình tạm đặt là:**Range(“A1:D6″).Select**  . Để kết thúc việc đặt tên, bạn nhập chuột ra một ô bất kỳ trên bảng tình mà không trùng với nút.



Bây giờ, chúng ta có thể test cái nút bằng cách click chuột trái vào chuột và bạn sẽ nhìn thấy vùng dữ liệu đã được chọn trong viền màu xanh (Màu đen đối với Office 2010 nhé ^^)

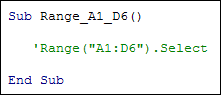


Thật tuyệt vời, bạn vừa viết một đoạn code Excel VBA để lựa chọn một vùng dữ liệu trên sheet hiện thời và tất cả có thể được kích hoạt thông qua một cái nút (button). Sau này, không chỉ với những dòng sub đơn giản mà bạn có thể gán những chương trình lớn của mình để toàn bộ có thể hoạt động chỉ với một hoặc nhiều hơn thế.

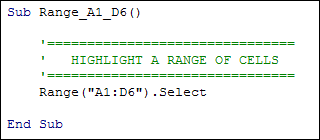
Ok, chúng ta quay trở lại trình biên dịch code (Alt + F11 – hoặc click biểu tượng Visual Basic trên thanh công cụ Developer). Viết một đoạn sub đơn giản sau:  (Nhớ xóa cái sub cũ đi nhé).

*Sub Range\_A1\_A6()*  
*Range(“A1:D6″).Select*  
*End sub*

Trong nó cũng cũng khá giống cái sub cũ nhỉ. Nhưng bạn hãy chịu khó viết xóa viết xóa nhiều lần. Điều này sẽ giúp bạn nhớ lâu hơn đó. Sau khi đã có dược đoạn code trên, bạn thử thêm một dấu nháy đơn vào trước đối tượng Range xem sao. Có vẻ như cái dòng đó như con tắc kè đổi sang màu xanh giống như cái hình dưới đây nhỉ:



Lý do nó đổi màu chỉ bởi nó có dấu nháy ở phía trước, nó là dấu nháy đơn. Các bạn có thấy một ‘dấu phẩy đặt lưng lửng’ thôi có sức ảnh hưởng như nào chưa ? — Tôi thích cách suy nghĩ của một số bạn khi cười đấy. ^\_\_^. Khi có một dấu nháy đơn ở đầu, thì VBA sẽ hiểu rằng đó là một chú thích (nghe quen quen, hình như nói ở đâu rồi hay sao ấy – thử tìm lại xem) . Và chương trình sẽ bỏ qua những dòng này . Nếu bạn chịu khó có thể tạo  một chú thích hoành tráng giống như bên dưới đây:



Chăm chỉ chèn những đoạn chú thích vào những dòng code là một thói quen tốt. Thật hữu ích khi bạn quay trở lại dự án của mình sau nhiều ngày bị gián đoạn. Nếu bạn không chú thích gì cả, rất có thể bạn sẽ không hiểu mình định làm gì với mớ code đó. Tin mình đi, đó là sự thật đấy. Không cần quá cầu kì nhưng cũng nên có những dòng chú thích ở những khóa quan trọng. Đó là phong cách lập trình chuyên nghiệp – Và với một phần mềm có mức độ code lớn như [phần mềm kế toán](http://www.h2account.com/phan-mem-ke-toan) H2 Account. Bọn mình vẫn thi thoảng chèn những dòng chú thích để phục vụ cho công tác kiểm tra, update sau này.

Trở lại với đoạn code ở trên. Chúng ta làm thế nào để làm chiếu đến vùng dữ liệu từ A1 đến D6. Rất đơn giản:

Range(“A1:D6″)

Một cách khác để tham chiếu một vùng tương tự như thế này:

Range(“A1″, “D6″).Select

Cả hai ví dụ trên đều làm một việc giống nhau, vậy chúng ta tìm hiểu đi tìm hiểu lại cái ví dụ này để làm gì nhỉ? Hai cái thực tế đều dùng theo ý bạn nhưng với cách thứ hai. Bạn có thể sử dụng một thứ gọi là: ActiveCell

**ActiveCell**

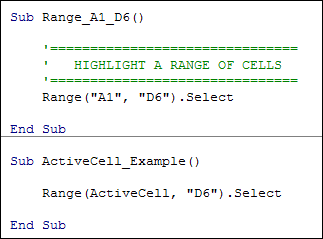
Cái này nghe cũng quen này, hình như chúng ta đã đề cập đến nó ở một bài nào đó rồi thì phải. ActiveCell chỉ đối tượng là ô (cell) mà con chuột của bạn đang hiện thời ở đó. ActiveCell là một từ viết liền nhé các bạn. Viết rời Active Cell là sẽ bị báo lỗi ngay. Vì bản chất ActiveCell là một ô, nó cũng giống như “A1″; hay “D6″ ở trên vì vậy nó cũng có thể được ghép vào đoạn code để người lập trình tận dụng được tính linh động của nó, chúng ta hãy thử xem một ví dụ sau:

**Sub ActiveCell\_Example()**

**Range(ActiveCell, “D6″).Select**

**End sub**

Hoàn thành thì trông nó giống như thế này



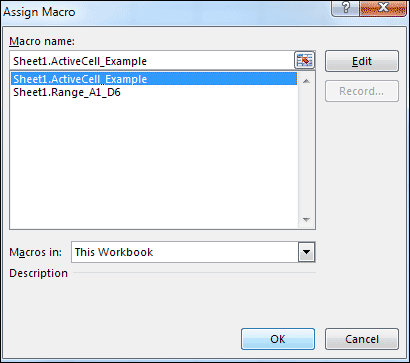
Nếu các bạn có thắc mắc về dòng kẻ đen xuất hiện thì đó chỉ là tính năng của Excel VBA, khi các bạn viết một sub mới thì VBA sẽ tự động thêm dòng kẻ này để giúp chúng ta dễ phân biệt giữa các sub với nhau. Một chương trình như phần mềm kế toán H2 Account là một tập hợp hơn vài trăm sub mà mỗi sub lại có một vai trò khác nhau để điều khiển hoạt động của phần mềm.

Trở lại vấn đề : ActiveCell chúng ta hãy so sánh sự thay đổi khi áp dụng:

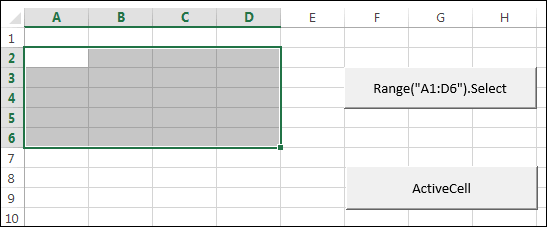
**Trước:**   **Range(“A1″, “D6″).Selec**t

**Sau:**       **Range(ActiveCell, “D6″).Select**

Thay vì sử dụng ô A1 là cố định, chúng ta sử dụng ActiveCell là ô sẽ thay đổi khi chúng ta di chuyển. Để thuận tiện, các bạn vẽ thêm một nút khác, tương tự như cái nút đầu tiên nhưng đặt nó ở bên dưới sau đó chọn vào Sheet1.ActiveCell\_Example  (Các bạn nhớ phải làm theo đúng các bước đặc biệt là các phần viết code để khi thả chuột tạo nút nó hiện lên macro như hình).  –> Sau đó ấn OK



Ngay sau đó bạn tiến hành đổi tên cái nút mới tạo thành: “ActiveCell”. Và giờ chúng ta sẽ thử kiểm tra khả năng hoạt động của nút bằng cách di chuyển con chuột tại các vị trí khác nhau trong vùng từ A1:D6. Thử đặt con chuột tại vị trí A2 và bấm nút có tên: “ActiveCell” bạn sẽ thấy Excel lựa chọn cho chúng ta vùng tương tự như hình bên dưới:



Tiếp tục click ra các vị trí khác và bấm nút ActiveCell để kiểm tra từ đó bạn sẽ hiểu hơn về sự biến hóa chỉ với những dòng code đơn giản trong Excel VBA.  Đương nhiên, để trở thành một chuyên gia, chúng ta cần phải nỗ lực không ngừng để nghiên cứu sâu thêm nhiều vấn đề khác trong VBA như cách thức xây dựng một chương trình, các khối lệnh logic, biến…nhưng hãy tạm thời để nó vào các bài sau. Bài tới chúng ta sẽ tìm hiểu về thuộc tính OFFSET, một thuộc tính rất hay được sử dụng khi lập trình VBA.

 Thay vì click ngay để vào học Bài 6. Mình có lời khuyên cho các bạn là hãy đọc lại thật kĩ bài 5 này, thực hành nhiều lần + thêm việc ôn lại kiến thức từ các bài khác nữa để có kiến thức nền tảng vững chắc về VBA.

***Mình tin rằng, các bạn sẽ thành công nếu có phương pháp học tập tốt!***

Tự học VBA – Bài 6: Thuộc tính Offset

Chào mừng các bạn đến với http://h2account.com – Chuyên trang [phần mềm kế toán](http://www.h2account.com/phan-mem-ke-toan)excel H2 Account . Chúng tôi xây dựng nên những bài học miễn phí này giúp các bạn dễ dàng tiếp cận hơn với một công cụ hỗ trợ đắc lực cho công việc của mình hàng ngày. Đó là chính là VBA. Không giống như các ngôn ngữ khác có thể mang lại cho bạn một công việc có thu nhập cao và ổn định. VBA là một ngôn ngữ không nhận được sự quan tâm của nhiều người. Tuy nhiên, đối với một số ngành nghề đặc thù ở Việt Nam hay phải sử dụng đến excel thì VBA là một thứ vũ khí vô cùng hiệu quả để các bạn chinh phục đỉnh cao nghề nghiệp bằng cách tự động hóa phần lớn các công việc vốn được coi là nhàm chán nếu chỉ thao tác đơn thuần trên excel.

Với những bạn đã học từ Bài 1 đến Bài 5 một cách chăm chỉ thì mình nghĩ rằng Bài 6 này cũng sẽ được các bạn chinh phục một cách dễ dàng bởi nó cung cấp cho các bạn nền tảng kiến thức hữu ích đối với một trong số những thuộc tính được sử dụng nhiều trong khi viết code đó là thuộc tính OFFSET.

**OFFSET là gì?**  – Dịch nghĩa tiếng Việt theo hướng liên quan đến excel thì đó là một sự lệch trục. Cắt nghĩa chữ lệch trục thì để sự lệch trục này diễn ra chúng ta phải có một cái trục ^^ và sự lệch của nó. Lệch có rất nhiều kiểu nhưng chắc hẳn các bạn cũng biết đến chiều ngang và dọc (Hàng và cột trong excel đó) ^\_\_^ ơ rê ka… vậy OFFSET khi xảy ra trong excel tức là có sự tác động để làm lệch trục theo hàng và cột.

Trục – Theo các bạn thì trong excel chúng ta có cái gì để làm trục, làm mốc…? ngoài ô và tập hợp các ô ra thì chúng ta còn có cái gì đâu! Chính xác…ô và tập hợp các ô nếu được chỉ định thì sẽ được quy làm trục khi chúng ta sử dụng thuộc tính OFFSET.

Hãy tư duy thật nhanh: –> Lựa chọn (tham chiếu) một ô, một vùng dữ liệu trong excel VBA thì làm thế nào?

range(A1).select

Không, không phải, hình như chúng ta đã quên cái gì đó…Chính xác phải là

**range(“A1“).select**

Ok, đó chính là cách tham chiếu tới ô A1 trong bảng tính hiện thời. Đừng quên cách tham chiếu tới một vùng dữ liệu như sau nhé:

**range(“A1:C3″).select**

Khi sử dụng thuộc tính Offset chúng ta sẽ phải làm như thế nào, các bạn hãy quan sát VD bên dưới:

**Range(“A1″).Offset(RowOffSet:=1, ColumnOffset:=1).Select**

Đoạn code trên sẽ cho chúng ta tham chiếu tới B2. Nhưng “trục” mà ta sử dụng là A1. Để sử dụng thuộc tính Offset, ngay sau khi dấu ngoặc đơn đóng đối tượng Range., chúng ta sử dụng dấu chấm, tiếp theo là OFFSET (các bạn viết thường hay viết hoa đều được). Sau Offset là một dấu ngoặc đơn chứa các giá trị quy ước của thuộc tính này đó là: RowOffSet (Độ lệch dòng) và ColumnOffset (Độ lệch cột). Giữa hai giá trị này được cách nhau bởi dấu phẩy.

Như vậy đoạn code trên có nghĩa như sau: Hãy lựa chọn vào vị trị được dịch 1 cột **sang phải** và 1 dòng **xuống dưới**so với ô A1. (Hãy suy nghĩ xem tại sao mình lại bôi đậm hai từ đó – câu trả lời nằm ở cuối bài)

Các bạn có thể viết code nhanh hơn bằng việc thay thế RowOffSet và ColumnOffset bằng luôn các giá trị cụ thể:

**Range(“A1″).Offset(1, 1).Select**

Hoặc lược bỏ đi thành phần cột nếu bạn thích:

**Range(“A1″).Offset(1).Select**

Với đoạn code này, các bạn đã loại bỏ đi thông số về độ lệch cột khi sử dụng thuộc tính Offset. Excel sẽ hiểu rằng bạn chỉ muốn dịch chuyển xuống một dòng so với mốc tiêu chuẩn là A1. Có nghĩa là excel sẽ lựa chọn cho bạn ô A2.

Một số bạn có phong cách khác là chỉ muốn dịch cột chứ không muốn dịch dòng vậy thì hãy quan sát ví dụ này:

**Range(“A1″).Offset(, 1 ).Select**

Thật thú vị, VBA rất hấp dẫn phải không nào, vị trí cột đã biến mất bỏ lại dòng bơ vơ một mình bên cạnh dấu phẩy. Excel sẽ hiểu rằng bạn muốn dịch chuyển 1 cột sang phải so với mốc tiêu chuẩn là A1. Có nghĩa là excel sẽ lựa chọn cho bạn ô B1.

Ngông hơn nữa, một số bạn muốn sử dụng dấu âm (-) để thể hiện khả năng chi tiêu vượt mức của mình

**Range(“B2″).Offset(-1, -1 ).Select**

Với đoạn code này, mốc tiêu chuẩn chúng ta lựa chọn lại là B2, và sự hiện diện hai giá trị dòng và cột trong thuộc tính Offset là (-1, -1). Khi đó, Excel sẽ dịch chuyển một cột**sang trái** và một dòng **lên trên** so với mốc tiêu chuẩn B2. Có nghĩa là excel sẽ lựa chọn cho bạn ô A1.

Nếu bạn chịu khó đầu tư, bạn sẽ tham chiếu lệch trục so với mốc tiêu chuẩn là một vùng dữ liệu chẳng hạn như sau:

**Range(“A1:C3″).Offset(1, 1).Select**

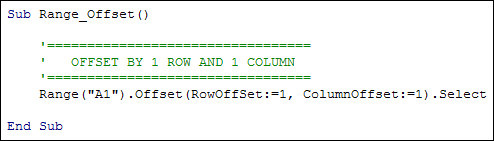
Với đoạn code trên, cả vùng dữ liệu A1:C3 sẽ được thuộc tính Offset dịch chuyển sang ngang 1 cột và xuống dưới 1 cột. Như vậy, A1 sẽ thành B2 và C3 sẽ thành D4. –> A1:C3 –> B2:D4.

Trăm nghe không bằng một thấy, nếu bạn đọc hết các đoạn trên mà vẫn mông lung thì hãy xóa sạch mọi dòng code đi và chúng ta cùng thử nghiệm với các nút trực tiếp trên bảng tính excel để trực quan hơn. Tạo nút như thế nào nhỉ? Nếu bạn quên thì thật đáng trách đó. Hãy quay ngay lại [bài 5](http://www.h2account.com/tu-hoc-vba-bai-5-tao-nut-trong-bang-tinh) để đọc lại thật kỹ và ghi nhớ nhé.

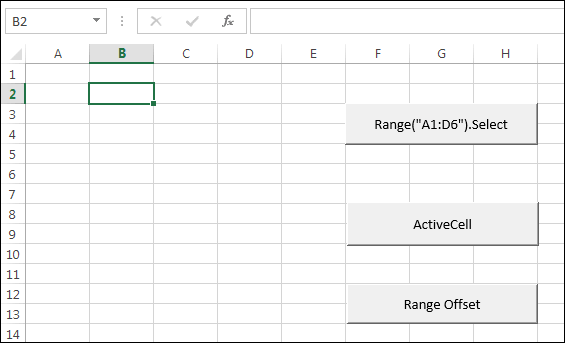
Bây giờ, các bạn trở lại vùng viết code và tạo một Sub với tên như sau: Range\_Offset với phần thân được code như dưới đây:

**Range(“A1″).Offset(RowOffSet:=1, ColumnOffset:=1).Select**

Khi hoàn thành thì cửa sổ viết code của bạn sẽ trông đại khái như thế này:



Quay lại với bảng tính excel, các bạn vẽ một nút chèn Macro Range\_Offset và đổi tên nó thành Range Offset rồi bấm cái rụp xem vùng chọn nó đi đến đâu? B2 đúng không? Ok



*(Trong hình các bạn có hai nút bên trên là do mình chụp luôn ở bài trước ^^)*

Tiếp theo, các bạn thử thay đổi 2 giá trị của thuộc tính Offset giống thế này:

**Range(“A1″).Offset(2, 2).Select**

Kiểm tra khả năng hoạt động của đoạn code đó với  cái nút Range Offset. Bây giờ thì Excel sẽ lựa chọn ô nào nhỉ?

Chưa dừng lại ở đó, hãy tiến hành thử với hàng loạt đoạn code sau cho mình. Với mỗi một đoạn code, hãy trở ra bảng tính excel và bấm nút rồi quan sát so sánh với ô mà excel chọn với ô A1 (độ lệch cột, độ lệch dòng) và Ơ rê ka nhé ^\_\_^

**Range(“A1″).Offset(3).Select**  
**Range(“A1″).Offset(, 2 ).Select**  
**Range(“B2″).Offset(-1, -1 ).Select**  
**Range(“A1:C3″).Offset(1, 1).Select**

Các cụ có câu, học đi đôi với hành đúng ko, giờ thì cho mình “củ hành” các bạn một chút nhé. Phải động não thì mới có nhiều nếp nhăn…khà khà.

**BÀI TẬP**  
Giả sử,

Bạn sử dụng giá trị cho đối tượng Range là ô A2. Làm cách nào bạn tới được ô C5 mà sử dụng thuộc tính Offset?

Với Range(“E8″), làm thế nào bạn dịch tới ô B3 mà sử dụng thuộc tính Offset?

Tại sao bạn không thử comment câu trả lời vào ngày dưới bài này để cho cả thế giới biết rằng bạn đã và đang tự học VBA trên website này nhỉ! Trước khi học sang thuộc tính tiếp theo, hãy hứa với mình rằng các bạn đã hiểu và nắm chắc cách sử dụng thuộc tính offset nhé.

Tự học VBA – Bài 7: Thuộc tính Resize

**Resize** dịch ra tiếng Việt là thay đổi kích thước.  Trong bài học này chúng ta sẽ tìm hiểu về thuộc tính **Resize**, ồ vậy thì bài học này chúng ta sẽ tìm cách giảm kích thước của một file excel xuống có đúng không nhỉ? Không, rất tiếc là chúng ta chưa tìm hiểu đến kĩ thuật đấy. **Resize** trong bài học này có nghĩa là **thay đổi vùng chọn**

Các bạn hãy nhớ lại ở bài học trước, chúng ta tìm hiểu về OFFSET. Đó là thuộc tính giúp chúng ta chọn một vùng dữ liệu từ một điểm mốc ban đầu. Nếu mốc ban đầu là một ô thì sử dụng Offset chúng ta sẽ chọn ra một ô khác. Nếu mốc ban đầu là một vùng dữ liệu chúng ta cũng sẽ được một vùng dữ liệu có số ô chiều ngang và dọc giống hệt vùng giữ liệu ban đầu. Offset đơn thuần chỉ là sự dịch chuyển đối tượng lệch trục.

Còn đối với Resize, chúng ta sẽ thay đổi vùng chọn ban đầu bằng cách thêm ô, hoặc cột cho nó kể từ vùng dữ liệu đầu tiên được chọn. Có vẻ hơi khó hiểu nhỉ, hãy cùng mình phân tích vị dụ bên dưới:

**Range(“A1″).Resize(RowSize:=2, ColumnSize:=2).Select**

Vẫn khởi đầu bằng Range(“A1″) và sau đó là thuộc tính Resize với các thông số: RowSize, ColumnSize –> Row là cột và Column là dòng. Như vậy có thể dịch câu lệnh trên như sau: Hãy lựa chọn một vùng dữ liệu bắt đầu từ ô A1 và có số dòng là 2 và số cột là 2. Từ ô A1 mà lại có số dòng là 2 tức là A1:A2, Số cột là 2 tức là A1:B1 . Ghép lại thì ta sẽ có vùng dữ liệu A1:B2. Đến đây các bạn đã thấy sự thay đổi của vùng dữ liệu chưa? Nếu không dùng Resize thì chúng ta chỉ chọn được một ô A1 với câu lệnh: Range(“A1″).Select. Nhưng sau khi dùng Resize thì chúng ta đã chọn được cả một vùng dữ liệu gồm 4 ô là: A1:B2 (A1, A2, B1, B2)

*Lưu ý:*

*–> Bạn nào không hiểu sao lại viết A1:B2 thì xem lại kiến thức excel căn bản nhé ^\_^*

–> *Offset thì là RowOffset = …, Resize là RowSize = …  Có sự sai khác nhất định, một thằng bê nguyên, một thằng bỏ chữ Re ở đầu đi. Nhớ nhé các bạn!*

–> *Giá trị thông số của thuộc tính Reszie không có giá trị âm.*

Chúng ta cũng có thể viết tắt các thông số của thuộc tính Resize như dòng code dưới đây:

**Range(“A1″).Resize(2, 2).Select**

Nếu bạn chỉ muốn điều chỉnh thêm cột mà không thêm dòng thì sử dụng code này:

**Range(“A1″).Resize(2).Select**

Và nếu bạn chỉ muốn điều chỉnh thêm dòng mà không muốn thêm cột thì làm như sau:

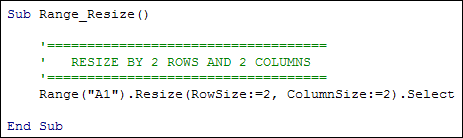
**Range(“A1″).Resize(, 2).Select**

Trăm nghe không bằng một thấy, chúng ta sẽ tiến hành thực hành trên vùng viết code trong VBA để các bạn có thể thấy rõ thuộc tính Resize hoạt động như thế nào.

Mở vùng viết code lên, gõ một sub đơn giản với tên: Range\_Resize. Ở giữa Sub và End Sub là dòng code bên dưới: (Lưu ý là nếu các bạn đang ở Sheet1 thì trong trinh biên dịch các bạn ấn vào Sheet1 để gõ nhé, ấn sang sheet khác thì báo lỗi đấy, bạn nào ăn chơi gõ vào Thisworkbook thì mở sheet nào cũng ok, nhưng khi chạy thì chỉ có sheet hiện hành có tác dụng):

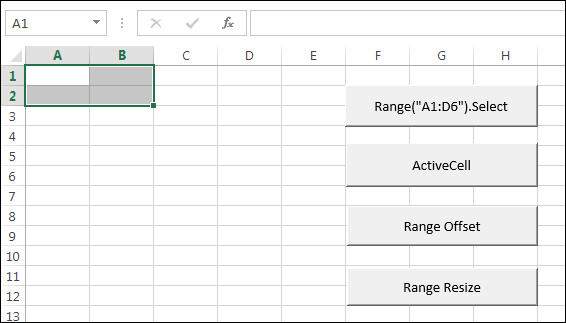
**Range(“A1″).Resize(RowSize:=2, ColumnSize:=2).Select**

Đoạn code xinh đẹp của bạn sẽ trông giống như thế này:



Quay trở lại cái sheet thân yêu của chúng ta ở các bài học trước, các bạn tạo nhanh một cái nút, khi hộp thoại hiện ra thì chọn Range\_Resize từ danh sách –> Click OK –> Đổi tên cái nút thành Range Resize

Để kiểm tra sự hoạt động của nút Range Resize, các bạn click vào ô A1 để làm ví trí bắt đầu. Sau đó click nút mới tạo và bạn có thể dễ dàng nhận thấy excel đã lựa chọn vùng dữ liệu từ ô A1 đến ô B2.



Thay thế đoạn code trong Sub Range\_Resize bằng các đoạn code dưới đây và bấm nút. Thực hành nhiều lần là cách hiệu quả giúp các bạn hiểu sâu, nhớ lâu về cách thức hoạt động của thuộc tính Resize:

Range(“A1″).Resize(3, 4).Select  
Range(“A1″).Resize(3).Select  
Range(“A1″).Resize(, 4).Select  
Range(“A1:A3″).Resize(, 2).Select

*Lưu ý rằng: ví dụ cuối cùng sử dụng vùng dữ liệu A1:A3 cho đối tượng Range. Khi đó thuộc tính Resize(,2) sẽ điều chỉnh vùng chọn thành A1:B3. Chúng ta cần đặc biệt ghi nhớ là thuộc tính Reszie luôn ưu tiên ô đầu tiên, trong****A1****:A3 thì****A1****được ưu tiên và lấy làm điểm mốc, các thay đổi cột, dòng có thể làm ảnh hưởng đến A3 chứ không ảnh hưởng đến A1.*

VD: Range(“A1:A4″).Resize(2,2).Select thì vùng lựa chọn sẽ là A1:B2  thay vì có 4 dòng như lúc đầu thì vùng chọn mới này chỉ còn 2 dòng.

Thú vị nữa, bạn có thể sử dụng kết hợp 2 thuộc tính Resize và Offset và khi đó dòng code sẽ trở nên khó hiểu hơn đôi chút.

**Range(“B1″).Offset(, -1).Resize(, 3).Select**

Ô được lựa chọn là B1, ô này bị thuộc tính Offset dịch chuyển sang trái một cột (do dấu trừ (-)). Như vậy B1 sẽ là A1. A1 tiếp tục chịu ảnh hưởng của thuộc tính Resize cộng thêm 3 cột nữa. Kết quả cuối cùng vùng được lựa chọn là A1:A3 . Quả thực rất thú vị đúng không các bạn.  Có ai thắc mắc là đảo ngược lại Resize trước, Offset sau thì sao? Thì các bạn tự thử khám phá và rút ra kinh nghiệm nhé.

**Bài tập**  
Giả sử ô được lựa chọn là A2 (Range(“A2″)), sử dụng Resize để lựa chọn A2:C2 –> Nhớ comment ở ngay dưới bài học này nhé các bạn!

OK, như vậy là chúng ta đã kết thúc **PHẦN 1: NHỮNG KIẾN THỨC CĂN BẢN.**thực sự là cũng không quá khó ngay cả với bạn mới bắt đầu chưa có nền tảng gì về lập trình. Sang phần tiếp theo, các bạn sẽ phải vận dụng nếp nhăn bộ não nhiều hơn, thử thách nhiều hơn và nếu qua được phần 2 các bạn mới thực sự hiểu về lập trình VBA ở mức độ nghiệp dư. Đầu xuôi thì đuôi sẽ lọt nên các bạn cứ yên tâm. Webiste chuyên về phần mềm kế toán excel H2 Acount sẽ luôn ở bên bạn, cùng bạn đi tới con đường chinh phục thành công VBA.

Tự học VBA – Bài 8: Khái niệm cơ bản về biến

Bắt đầu sang bài 8, chúng ta sẽ học các khái niệm chuyên môn về lập trình hơn, nó sẽ khá trừu tượng đòi hỏi mỗi người học cần tập trung, ghi nhớ và thực hành liên tục.  Vì vậy, trang web chuyên về [***phần mềm kế toán excel***](http://www.h2account.com/) H2 Account rất hi vọng các bạn đủ bình tĩnh, tự tin vào thành công đang chờ đợi mình ở phía trước.

**Chúng ta bắt đầu nào!**

Trong lập trình, chúng ta phải thiết lập các bước theo một chu trình nhất định để máy tính có thể hiểu và thực thi nhiệm vụ. Và bước đầu tiên của phần lớn các ngôn ngữ lập trình đó là khai báo biến. Hãy hình dùng bạn khai báo biến giống như bạn làm toán vậy:

**Giả sử** **A** là **chiều dài của cạnh hình vuông**.

Việc bạn khai báo biến đồng nghĩa với việc bạn lưu giữ chúng trong bộ nhớ, chẳng hạn như số, chuỗi ký tự. Tại sao phải lưu giữ nhỉ? Tại vì bạn cần sử dụng lại chúng ở các đoạn code sau đó mà tác dụng của nó cũng tương tự. Các biến thì đều có tên do chính bạn đặt. Theo như ví dụ trên, thì **A** là biến, một số bạn không thích **A** có thể dùng B hoặc AB …nói chung bất cứ tên gì mà thuận tiện với bạn trong khi bạn lập trình.

Trong ngôn ngữ lập trình VBA, bạn cũng cần khai báo cẩn thận các biến mình sẽ sử dụng (*Thực tế thì VBA không quá khắt khe trong việc đặt biến nhưng các bạn nên tập thói quen này để biết đâu sau này bạn học ngôn ngữ cao hơn sẽ không quá phức tạp nữa. Từ bài này trở đi, trong hầu hết các ví dụ, bài thực hành thì chúng ta đều vận dụng theo trình tự đặt biến đầu tiên*). Vì ngôn ngữ lập trình nên chúng ta phải thuộc từ mới của ngôn ngữ đó. Bắt đầu bởi từ Dim . Dim có nghĩa giống như giả sử vậy.  Hãy cùng mình phân tích một cách đặt biến cơ bản:

**Dim** **MyNumber** As **integer**

Chú ý đến màu sắc mình sử dụng, các bạn có liên tưởng được điều gì không? Dim là từ khóa bắt buộc, MyNumber thì tùy ý bạn, As là từ khóa bắt buộc integer là dạng biến. Dạng biến là gì? integer là số nguyên, mà số nguyên thì không thể đại diện hết cho các biến được vì vậy người ta phải đẻ thêm ra: string là chuỗi ký tự. string và integer chính là các dạng biến.

Vậy có tất cả bao nhiêu dạng biến? Trả lời rằng có khá nhiều dạng biến và chúng ta thường sử dụng một số dạng biến quen thuộc mà thôi. Trong phạm vi của bài học này: các bạn chỉ cần ghi nhớ cho mình 2 dạng biến: integer và string. OK?

Dịch nghĩa tiếng Việt thì đoạn code trên có nghĩa là: **Giả sử** **MyNumber** là **số nguyên**

Tiếp theo, nếu chỉ giả sử như vậy thôi thì có vẻ như không có cái gì được lưu trữ bên trong biến MyNunber. Nói MyNumber là số nguyên thì quá chung chung và nếu như các bạn muốn gán nó bằng một giá trị cụ thể nào đó (hằng số) thì các bạn sử dụng dấu “=”. Theo thuật ngữ chuyên môn thì các bạn vừa thực hiện việc gán một toán tử. Cụ thể, bạn sẽ gán như sau:

**MyNumber = 10**

Rất đơn giản phải không bạn? Chỉ cần lấy tên biến đặt trước dấu = và sau đó là giá trị cụ thể nào đó mà bạn muốn gán. Đáng tiếc thay bạn không thể đặt biến và gán giá trị cho nó trên cùng một dòng (Điều mà bạn chỉ có thể làm trong Visual Basic .NET mà không làm được trong Visual Basic for Applications. (VBA). Tức là bạn không được phép làm như sau:

**Dim MyNumber As Integer = 10   (SAI)**

**Dim MyNumber As integer  (ĐÚNG)**

**Mynumber = 10** (ĐÚNG)

**TÊN BIẾN**

Trong bài, mình có nhắc đến khá nhiều về tên miền MyNumber. Cũng như tên Sub, khi đặt tên biến người viết code VBA cần phải tuân thủ những quy định bắt buộc như sau:

* Bạn không thể bắt đầu tên biến bằng một số
* Bạn không thể dùng dấu cách khi đặt tên biến
* Bạn không thể sử dụng các ký tự đặc biệt  như:  !, %, �, #, $

Và đây là một số cách đặt tên biến mà bạn có thể xài OK

**MyVariable**  
**My\_Variable**  
**myvariable2**

Và một số cách dưới đây sẽ khiến VBA báo lỗi:

**2MyVariable  (Sử dụng số đầu tên biến)**  
**My Variable (Có khoảng cách trong tên biến)**  
**$myvariable (Sử dụng ký tự đặc biệt trong tên biến)**

Nhắc lại: Bạn nên đặt tên biến là cái gì đó có thể khiến bạn dễ nhớ. Chẳng hạn bạn định đặt thuế suất là biến thì có thể viết như sau: ThueSuat . Rõ ràng sẽ dễ hình dung rất nhiều nếu bạn viết tắt là TS hoặc một ký tự ABC nào đó.

Trong bài tiếp theo, các bạn sẽ thực hành về việc sử dụng biến trong Excel VBA.

Tự học VBA – Bài 9: Thực hành về biến

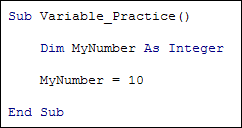
Trước khi vào bài học này, chúng ta sẽ cùng liệt kê lại một số kiến thức mà ở bài trước đã đề cập đến như khái niệm cơ bản về biến, cách gán toán tử (hằng số) cho biến, thêm một sheet mới, hay thao tác với “Ví thạch sanh” – thanh công cụ Developer dành cho người [***lập trình VBA***](http://www.h2account.com/tu-hoc-vba-trong-excel).  Mình hi vọng, khi đọc đến đâu, các bạn cũng đều nhớ lại được các kiến thức đã được học. Thường xuyên suy nghĩ và tìm tòi là chìa khóa giúp con người thành công!

Bây giờ, chúng ta sẽ đi vào [**Bài 9: Thực hành về biến**](http://www.h2account.com/tu-hoc-vba-bai-9-thuc-hanh-ve-bien) để các bạn có thể nhìn trực quan hơn về việc sử dụng biến trong lập trình VBA. Trong Sheet1 của trình biên dịch VBA các bạn tạo một Sub mới với tên: Variable\_Practice,  đi cùng là phần khai báo biến như sau:

**Dim MyNumber As Integer**

**MyNumber = 10**

Khi đó, đoạn code của bạn sẽ trông giống như thế này:



 OK. chúng ta vừa khai báo một biến tên là MyNumber và gán co nó một giá trị bằng 10 . Nhưng nếu chỉ đơn giản như thế thì thực lòng mà nói, đoạn code trên chẳng để làm gì cả. Thường thì người lập trình sẽ tiếp tục suy nghĩ các bài toán mình sẽ phải giải quyết và nếu như có một yêu cầu là các bạn hãy viết một chương trình mà khi ấn nút sẽ tự động điền vào ô A1 giá trị bằng 10.

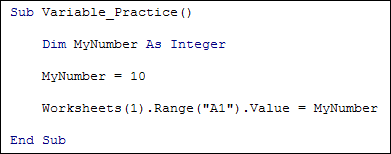
Quá đơn giản:—>  Worksheets(1).Range(“A1″).Value = 10

Thật tuyệt vời nếu ngay khi đọc xong đề bài bạn hình dung ra được dòng code trên. Và còn tuyệt vời hơn nữa nếu bạn cũng đang liên tưởng đến bài chúng ta đang nói đến chính là: thực hành biến để biến đoạn code trên thành:

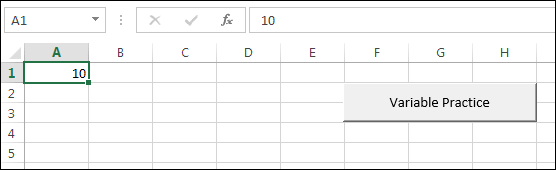
**Worksheets(1).Range(“A1″).Value = MyNumber**

Thay vì gõ số, chúng ta hoàn toàn có thể thay thế bằng một biến đã được khai báo và gán giá trị trước đó. VBA sẽ thấy được biến MyNumber và tiến hành lấy giá trị đã được lưu trong nó. Điều này nếu chỉ đơn thuần là 10 hoặc MyNumber thì sẽ rất vô nghĩa, nó chỉ thực sự cho thầy tầm quan trọng khi đặt trong một chương trình lớn khi mà biến trả về là kết quả của các thao tác người dùng chứ không phải được mặc định nữa.  Khi đó từ “Biến” sẽ đúng như tên của nó “Biến đổi”. Biến sẽ thay đổi tùy theo thao tác và ứng với những thay đổi đó, kết hợp với các dòng code chúng ta sẽ nhận được các kết quả khác nhau thỏa mãn nhu cầu tính toán.

Và sau khi thêm được dòng code đã đề cập ở trên, chương trình bạn viết sẽ trông giống như thế này:

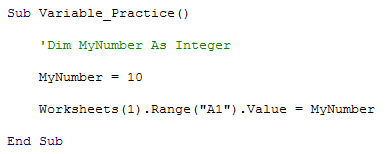


Vẫn thực hiện bài cũ – Tạo nút mới bên ngoài sheet để quan sát tận mắt. Khi ấn vào nút Variable Practice các bạn dễ dàng quan sát thấy số 10 đã được tạo ra ở ô A1.



Quay trở lại trình biên dịch – phần viết code, chúng ta thử nghịch một chút bằng cách thêm một dấu nháy đơn vào đầu đoạn khai báo biến: **Dim MyNumber As Integer**

Khi đó đoạn code sẽ trông giống như thế này



Hãy nhớ việc các bạn vừa làm dù chỉ thêm một dấu nháy nhưng nó rất quan trọng vì nó biến sự khai báo biến của chúng ta thành một đoạn ghi chú và đương nhiên VBA sẽ bỏ qua phần này. Và đoạn code mà VBA sẽ thi hành giờ chỉ còn:

**MyNumber = 10**

**Worksheets(1).Range(“A1″).Value = MyNumber**

**Mình có một câu hỏi**: **Đoạn code trên sẽ chạy bình thường hay báo lỗi?**

Chúng ta sẽ tìm câu trả lời bằng cách quay trở lại với bảng tính Excel và click vào nút**Variable Practice**. Ồ, nó vẫn ok như chẳng có gì xảy ra cả. Tại sao lại như vậy? Chúng ta đã không đặt biến thế mà VBA vẫn hiểu được, chuyện gì đang xảy ra vậy nhỉ?

Thực tế, bạn có thể đặt một biến như thế này:

**Tên biến = giá trị nào đó**

Nói một cách khác, bạn không thực sự cần khai báo biến thông qua cú pháp: **Dim**tên\_biến **As** integer. Bạn có thể bỏ qua nó tuy nhiên mình không khuyến kích điều đó bởi xét cho cùng vì VBA là ngôn ngữ lập trình thấp cấp và khá đơn giản nên có thể việc khai báo biến không thực sự quá quan trọng nhưng nó là một thói quen không hề tốt và với việc lười khai báo biến sẽ kéo theo rất nhiều hệ quả đi kèm như khó kiểm soát code, chương trình viết ra sẽ chạy ì ạch, hao tổn tài nguyên vân vân và vân vân… ^\_^

Hãy bắt đầu trở thành người lập trình VBA chuyên nghiệp bằng cách đặt biến bất cứ khi nào có thể với từ khóa Dim

**MỞ RỘNG**  
Xóa bỏ dấu nháy đơn trước từ khóa Dim ở phần trước và thay thế A1 bằng A1:A10. Sau đó trở lại bảng tính Excel và tiến hành chạy thử code của bạn. Điều gì sẽ xảy ra?  
Trong một chương trình, bạn có thể đặt nhiều biến hơn tùy thuộc vào nhu cầu thực tế, hãy thử xem đoạn code bên dưới:

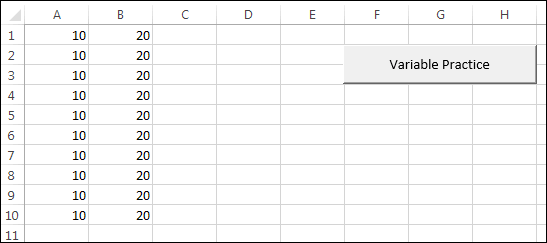
Dim MyNumber As Integer  
**Dim MyOtherNumber As Integer**

MyNumber = 10  
**MyOtherNumber = 20**

Worksheets(1).Range(“A1:A10″).Value = MyNumber  
**Worksheets(1).Range(“B1:B10″).Value = MyOtherNumber**

Chúng ta vừa đặt một biến khác có kiểu biến là số nguyên (integer) và tạm gọi nó là: MyOtherNumber. Và gán cho nó một hằng số là 20.

Khi click vào nút Variable Practice thì các bạn sẽ thấy:



Không hề khó phải không nào? Đừng quên cú pháp Dim Tên\_Biến As Kiểu biến và nhớ vận dụng nó mọi lúc có thể khi phải sử dụng đến biến nhé. Trong bài tiếp theo, chúng ta sẽ tìm hiểu về Option Explicit – Nghe có vẻ mơ hồ rồi đây…

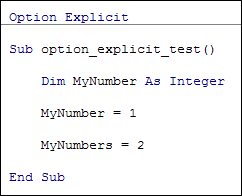
Tự học VBA – Bài 10: Câu lệnh Option Explicit

Trong lập trình nói chung và [lập trình VBA](http://www.h2account.com/he-thong-bai-hoc-vba-trong-excel) nói riêng chúng ta phải thao tác với rất nhiều biến, khi sử dụng nhiều biến như vậy thì lại rất dễ mắc sai sót. Hiểu được điều này, anh Bill đã tạo ra một câu lệnh giúp chúng ta giảm thiếu tối đa những sai sót trong quá trình sử dụng biến trong lập trình VBA. Câu lệnh đó chính là nội dung của bài học ngày hôm nay:

**Option Explicit**

Option dịch là ra là Tùy chọn, còn Explicit tức là minh bạch, rõ ràng. Có vẻ như câu lệnh này mang hàm ý làm rõ ràng những đám biến rối như tơ vò của những bạn học lập trình mà chưa chịu xây dựng thói quen cẩn thận.

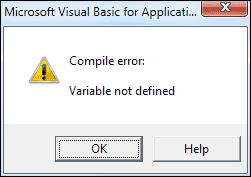
Trước tiên, để xài được câu lệnh này thì các bạn gõ: “**Option Explicit**” lên trên cùng, ngay trước khi viết các đoạn sub. Điều này giống như chúng ta vừa ra lệnh cho VBA: Nè, hãy check thật cẩn thận các biến của ta, rõ chưa! ^\_\_^ Hãy cùng mình tìm hiểu qua đoạn code ví dụ bên dưới:  (*Lưu ý: khi các bạn vừa gõ Option Explict xong và xuống dòng thì không thấy cái dòng gạch đen phân cách đâu, phải khi các bạn viết xong cặp Sub – End sub thì nó mới xuất hiện*)



Theo như đoạn code trên, chúng ta có câu lệnh Option Explicit đặt trên cùng, Đoạn Sub có một biến được khai báo thông qua từ koas Dim đó là: MyNumber . Tiếp theo chúng ta gán giá trị cho biến này là = 1 . Và các bạn hãy chú ý đến dòng bên dưới nữa:

**MyNumbers = 2**

Cứ coi như mình vô tình đánh nhầm MyNumber thành MyNumbers và trong trường hợp không sử dụng câu lệnh Option Explicit ở trên cùng thì VBA sẽ chạy đoạn code trên và thiết lập MyNumbers như một biến mới (chúng ta đã biết rằng trong VBA việc khai báo biến đôi khi là không bắt buộc). Tuy nhiên, việc viết nhầm như thế này, sẽ là nền tảng của những lỗi lầm khiến người viết code mất nhiều thời gian tìm kiếm, sửa chữa. Trở lại việc sử dụng câu lênh **Option Explicit** thì nếu như bạn lỡ có viết nhầm và chạy đoạn code đó thì VBA sẽ báo lỗi như thế này:

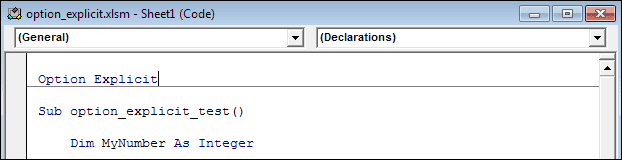


Thông báo lỗi trên nói cho chúng ta biết rằng có một biết đã không được khai báo trong mớ code bạn viết. Ở đây phải hiểu là biến đó không được khai báo bằng từ khóa Dim (Nên khắc cốt ghi tâm là trong VBA không có từ khóa nào khác ngoài từ Dim này để khai báo biến nhé các bạn)

Ok, vậy đến đây bạn đã hiểu được công dụng của câu lệnh **Option Explicit** chưa? Không dùng nó không chết ai nhưng dùng nó sẽ giúp chúng ta đỡ bị nhầm. Nó cũng là một thói quen tốt giống như việc chúng ta ghi chú các dòng code của mình vậy.

**General Declarations**

Nếu bạn nhìn kĩ bên trên cửa sổ viết code (trên cả Option Explicit)  bạn sẽ thấy có hai danh sách thả xuống:



Danh sách đầu tiên được gọi là **General** (Tạm dịch là tổng quát). Danh sách thứ hai được gọi là **Declarations (**Tạm dịch là khai báo) . Cái danh sách thứ hai nếu danh sách thứ nhất là General thì  nó sẽ hiển thị **Declarations** hoặc danh sách các Sub có trong cửa sổ viết code hiện thời. Nếu bạn chuyển **General** sang Worksheet thì ngay lập tức bên Declarations sẽ chuyển sang một list danh sách tạm gọi là danh sách các sự kiện. Nó là một phần khá phức tạp và sẽ được đề cập trong các chuyên đề chuyên sâu về VBA.

Bạn có thể thiết lập các biến tại khu vực Generral Declarations này, khi đó các biến sẽ được nhìn thấy ở mọi cửa sổ viết code. Còn bình thường, các biến chỉ có tác dụng tại Sub hoặc Function mà bạn thiết lập. (Có thuật ngữ mới Function chúng ta sẽ tìm hiểu sau nhé). Chẳng hạn: Biến MyNumber chỉ có thể được truy cập từ sub option\_explicit\_test . Nếu chúng ta có những sub khác trên cửa sổ viết code này thì biến MyNumber không thể sử dụng chung cho các sub đó. Tuy nhiên nếu chúng ta dịch chuyển dòng Dim MyNumber As integer lên khu vực General Declarations thì các sub khác dễ dàng xài lại được biến MyNumber.

Các bạn sẽ có những bài thực hành với việc thiết lập biến trong General Declarations ở các bài học chuyên sâu sau, còn phần tới chúng ta sẽ tìm hiểu về phép cộng và trừ trong VBA.

Tự học VBA – Bài 11: Phép cộng và phép trừ

**Sử dụng biến để cộng và trừ trong**[**Excel VBA**](http://www.h2account.com/he-thong-bai-hoc-vba-trong-excel)

Khi bạn lưu trữ các số bên trong các biến, ngoài việc xuất nó ra ngoài một ô đơn giản như trong các bài học trước chúng ta đã làm thì còn có thể thực hiện các phép toán đại số đối với các biến đó. Thực tế, đối với một chương trình lớn như [***phần mềm kế toán excel***](http://www.h2account.com/) H2 Account, việc sử dụng biến rất đa dạng và ứng với từng hoàn cảnh, nhiệm vụ mà bọn mình lập trình theo các hướng khác nhau. Chúng ta chỉ cần hiểu rằng việc sử dụng biến là cực kì đa dạng, phức tạp và để từng bước hiểu cách sử dụng biến thì chúng ta tập làm quen với các phép toán đơn giản, không đòi hỏi quá nhiều tư duy. Ví dụ như phép cộng.

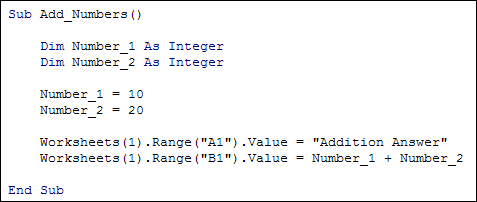
Để bắt đầu, chúng ta cũng sẽ khởi động excel, mở vào trình biên dịch VBA, chọn vào Sheet1 và cửa sổ code xuất hiện các bạn sẽ tạo một Sub và đặt tên nó là Add\_Numbers. Bên trong cặp Sub và End Sub chúng ta code như sau:

**Dim Number\_1 As Integer**  
**Dim Number\_2 As Integer**

**Number\_1 = 10**  
**Number\_2 = 20**

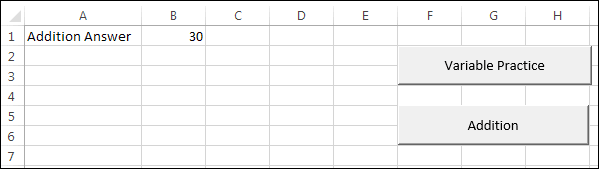
**Worksheets(1).Range(“A1″).Value = “Addition Answer”**  
**Worksheets(1).Range(“B1″).Value = Number\_1 + Number\_2**

Đoạn code của các bạn sẽ trông giống như thế này:



Rất đơn giản với việc khai báo hai biến Number\_1 và Number\_2, gán giá trị cho hai biến và sử dụng đối tượng Range cùng thuộc tính Value.

Trở lại với cửa sổ làm việc của Excel, các bạn tạo cho mình một nút đơn giản với tên: Addition. Nếu các bạn không biết cách tạo nút và gán Macro thế nào thì thật đáng trách, hãy quay trở lại bài 5 để học lại thật cẩn thận nhé. Tạo xong các bạn bấm vào nút rồi quan sát sẽ thấy tại ô A1 chúng ta có dòng chữ “Addition Answer” và tại ô B1 chúng ta có số: 30. 30 chính là kết quả của việc cộng hai biến: Number\_1 và Number\_2. (*Lưu ý: các bạn sẽ phải mở rộng code A để dòng chữ không bị mất nhé, thực ra chúng ta có thể code thêm vài dòng nữa để dòng A tự mở rộng nhưng mình sẽ không nói ra ở bài học này mà để các bạn tự suy nghĩ*).



Việc đặt dòng chữ vào ô A1 được thực hiện qua dòng code sau:

**Worksheets(1).Range(“A1″).Value = “Addition Answer”**

Các bạn lưu ý việc chèn chuỗi ký tự trong VBA thường được sử dụng trong dấu ngoặc kép “”. khá giống với trong các công thức excel. Và chúng ta khi code cũng phải tuyệt đối tuân thủ quy tắc này nếu không sẽ phát sinh lỗi ngay.

Việc cộng hai biến được thực hiện qua dòng code sau:

**Worksheets(1).Range(“B1″).Value = Number\_1 + Number\_2**

Bên phải dấu bằng ta có:

**Number\_1 + Number\_2**

Cũng không có gì phải giải thích nhiều ở đây, đơn giản chỉ là gõ tên hai biến và cộng chúng lại với nhau. Đương nhiên, bạn cũng được phép cộng liên tiếp nhiều biến một lúc chẳng hạn như:

**Worksheets(1).Range(“B1″).Value = Number\_1 + Number\_2 + Number\_3**

Và thực hiện phép cộng biến với một hằng số không phải là biến. Khi đó nó giống như thế này:

**Worksheets(1).Range(“B1″).Value = Number\_1 + Number\_2 + 30**

Hoặc chẳng cân dùng tới biến mà thực hiện việc cộng trực tiếp:

**Worksheets(1).Range(“B1″).Value = 10 + 20 + 30**

Chúng ta cũng có thể lưu trữ cả chuỗi phép cộng trong một biến duy nhất:

**Number\_1 = 10 + 20 + 30**  
**Worksheets(1).Range(“B1″).Value = Number\_1**

**Subtraction – Phép trừ**

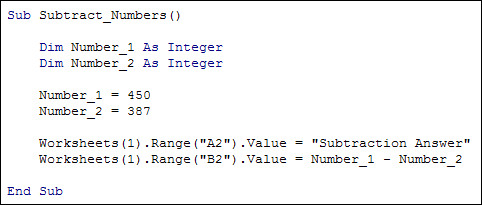
Trong ngôn ngữ lập trình VBA, phép trừ được thực hiện bằng việc sử dụng dấu trừ (-). Phép cộng đơn giản như vậy thì phép trừ cũng chẳng có gì phức tạp hơn, trở lại với cửa sổ code của bạn, chúng ta tạo một Sub khác với tên gọi là: Subtract\_Number. Và chèn vào giữa cặp Sub và End Sub đoạn code sau:

**Dim Number\_1 As Integer**  
**Dim Number\_2 As Integer**

**Number\_1 = 450**  
**Number\_2 = 387**

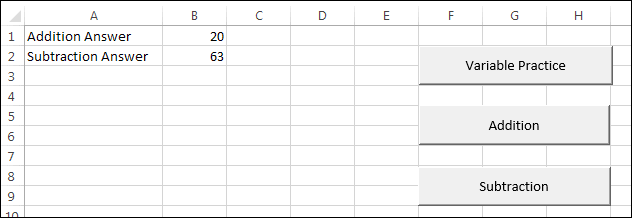
**Worksheets(1).Range(“A2″).Value = “Subtraction Answer”**  
**Worksheets(1).Range(“B2″).Value = Number\_1 – Number\_2**

Khi đó đoạn code tương tự như thế này:



Chú ý rằng: Chúng ta vừa sử dụng hai biến lặp lại là: Number\_1 và Number\_2. Điều này hoàn toàn có thể chấp nhận được bởi cả hai biến đều nằm trong cặp Sub và End Sub. Chúng ta chỉ gặp lỗi khi sử dụng lặp hai biến này ở cùng với Sub Add\_Numbers đã tạo ở phép cộng. Như vậy, nếu ở hai sub khác nhau chúng ta có thể sử dụng lặp lại tên biến và gán các giá trị hoàn toàn khác nhau cho biến. (Điều này được gọi là phạm vi biến).

Trở lại với bảng tính excel hiện thời, các bạn tạo một nút mới với tên gọi: Subtraction. và kiểm tra sự hoạt động của nút bằng cách bấm vào nó



Hoàn toàn tương tự như phép cộng, với phép trừ ở trên, ta khai báo hai biến và lưu trữ hai giá trị: 450 và 387 trong hai biến đó. Và phép trừ được thực hiện qua dòng code sau:

**Worksheets(1).Range(“B2″).Value = Number\_1 – Number\_2**

Một số cách ghi phép trừ khác mà bạn có thể tham khảo:

**Number\_1 – Number\_2 – Number\_3**  
**Number\_1 – 10**  
**300 – 200**

Bạn cũng có thể kết hợp giữa phép cộng và phép trừ như ví dụ dưới đây:

**Dim Number\_1 As Integer**  
**Dim Number\_2 As Integer**  
**Dim Number\_3 As Integer**  
**Dim Answer As Integer**

**Number\_1 = 50**  
**Number\_2 = 40**  
**Number\_3 = 30**

**Answer = Number\_1 + Number\_2 – Number\_3**

**Worksheets(1).Range(“A2″).Value = “Subtraction Answer”**  
**Worksheets(1).Range(“B2″).Value = Answer**

Chúng ta hãy để ý đến dòng code:

**Worksheets(1).Range(“B2″).Value = Answer**

Việc sử dụng Answer là một biến cho chúng ta thấy sự đa dạng của VBA, khi muốn nhận giá trị nào đó thì ngoài việc thao tác trực tiếp với biến tại dòng xuất giá trị, chúng ta có thể gán một biến chung và thao tác trên biến chung đó. Khi cần kết quả ta chỉ cần đưa biến chung vào dòng giá trị là xong. Điều này thực sự có ý nghĩa khi bạn phải sử dụng lại nhiều lần kết quả mà không muốn phải lặp lại các phép tính quá nhiều.

Trong bài học tới, mình và các bạn sẽ cùng tìm hiểu về phép nhân và phép chia trong VBA nhé.

Tự học VBA – Bài 12: Phép nhân và phép chia

**Sử dụng biến cho phép nhân và chia trong**[**Excel VBA**](http://www.h2account.com/he-thong-bai-hoc-vba-trong-excel)

Website chuyên về [***phần mềm kế toán excel***](http://www.h2account.com/) H2 Account xin chào các bạn! Ở bài học trước, chúng ta đã sử dụng biến để thực hiện phép cộng và phép trừ trong VBA, và để hoàn thiện 4 phép toán học căn bản thì trong bài học này chúng ta sẽ học cách sử dụng biến cho phép nhân và phép chia.

**PHÉP NHÂN**

Trong ngôn ngữ lập trình, dấu của phép nhân là dấu sao (\*). Và bây giờ nhiệm vụ là phải nhân 10 với 5 trong VBA.

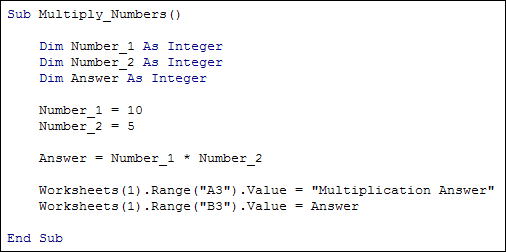
**Dim Number\_1 As Integer**  
**Dim Number\_2 As Integer**  
**Dim Answer As Integer**

**Number\_1 = 10**  
**Number\_2 = 5**

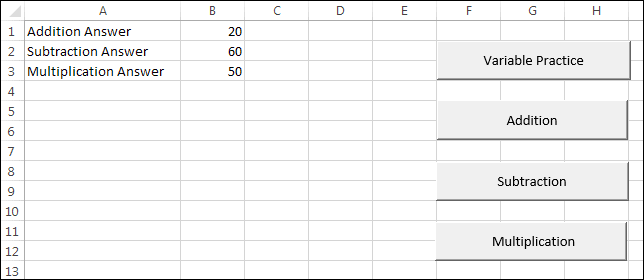
**Answer = Number\_1 \* Number\_2**

**Worksheets(1).Range(“A3″).Value = “Multiplication Answer”**  
**Worksheets(1).Range(“B3″).Value = Answer**

Không phải nói quá nhiều bởi nó rất giống hai phép toán chúng ta đã học ở bài trước. Các bạn chỉ cần đặt đoạn code trên bên trong cặp sub mà chúng ta tạm đặt tên là: Multiply\_Numbers



Một lần nữa, bạn lại làm nhiệm vụ quen thuộc đó là tạo một cái nút đặt tên là Multiplication (Phép nhân) và chạy nó. (*Đến đây mình cũng xin lưu ý với các bạn là trong bài học chúng ta liên tục sử dụng các nút tên tiếng Anh là để các bạn làm quen với ngoại ngữ này dần dần, bởi nếu không có tiếng Anh các bạn sẽ không thể học tốt được lập trình nói chung và VBA nói riêng*)



Hoàn toàn tương tự như phép cộng và phép trừ, bạn có thể sử dụng nhiều hơn hai số được lưu trữ trong các biến để thực hiện các phép tính toán của mình. Và đây là một số ví dụ:

**Answer = Number\_1 \* 10**  
**Answer = Number\_1 \* Number\_2 \* Number\_3**  
**Answer = Number\_1 \* Number\_2 \* 10**

Bạn cũng có thể kết hợp các phép toán đại số với nhau tương tự như khi làm toán hồi nhỏ. Và VBA cũng phải tuân thủ theo đúng nguyên tắc mà chúng ta ai cũng được học: Nhân chia trước, cộng trừ sau. Và trong trường hợp cần thiết các bạn cũng có thể sử dụng dấu ngoặc đơn để điều khiển phép toán của mình cho phù hợp.

**Answer = 10 \* (2 + 5)**

Nhìn chung là vẫn rất dễ đúng không nhỉ? Từ dễ tới khó và rất khó, bộ não của các bạn được trải nghiệm một cách từ từ, không vội vàng sẽ giúp nhớ lâu, nhớ sâu. Mỗi một viên gạch nhỏ ngày hôm nay sẽ giúp các bạn xây dựng một tòa tháp vươn tới đỉnh cao của sự thành công đó.

**PHÉP CHIA**

Ký hiệu được sử dụng khi bạn muốn chia số này cho số khác là ký hiệu dấu gạch chéo từ phải sang trái (/). Tuy nhiên ông VBA vẫn chấp nhận cho chúng ta sử dụng dấu gạch chéo từ trái qua phải (\) mà không hiển thị thông báo lỗi nào. Tuy nhiên, khuyến cáo là khi lập trình ta cứ sử dụng dấu gạch chéo chuẩn (/) để tránh nhầm lần với đường dẫn thư mục.

Chúng ta thực hành phép chia qua đoạn code bên dưới với tên sub là: Divide\_Numbers

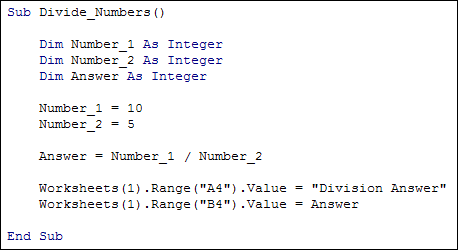
**Dim Number\_1 As Integer**  
**Dim Number\_2 As Integer**  
**Dim Answer As Integer**

**Number\_1 = 10**  
**Number\_2 = 5**

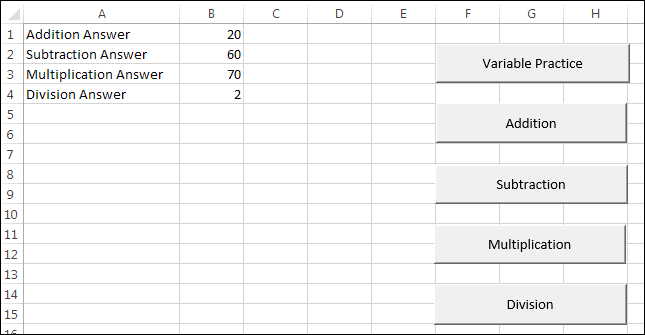
**Answer = Number\_1 / Number\_2**

**Worksheets(1).Range(“A4″).Value = “Division Answer”**  
**Worksheets(1).Range(“B4″).Value = Answer**

Khi đó đoạn code của bạn sẽ trông giống như thế này:



Trở lại excel –> Tạo nút (Division) –> Click và quan sát



Trở lại với trình viết cocde và thay thế biến Number\_2 có giá trị từ 5 thành 4.

**Number\_1 = 10**  
**Number\_2 = 4**

Tức là bây giờ, ta thực hiện phép chia 10 cho 4. Trở lại với Excel và click vào nút Division. Bạn nhận được kết quả là gì? Thay vì nhận được kết quả là 2.5 thì bạn lại nhận được kết quả là 2. Nguyên nhân là do VBA tự động bỏ đi phần .5 là do chúng ta đã khai báo biến là số nguyên (integer) ở ngay phần đầu của sub. Mà số nguyên thì không thể có phần chấm đi theo sau được.  Bởi vậy mới thấy được sự phức tạp của biến và trong từng trường hợp cụ thể, người lập trình cần xác định chính xác xem biến mình mong muốn nó thuộc dạng nào để trả về kết quả chính xác.  Bây giờ, bạn thay đổi đoạn code trong phép chia thành thế này:

**Dim Number\_1 As Integer**  
**Dim Number\_2 As Integer**  
**Dim Number\_3 As Integer**  
**Dim Answer As Integer**

**Number\_1 = 8**  
**Number\_2 = 8**  
**Number\_3 = 4**

**Answer = Number\_1 + Number\_2 / Number\_3**

**Worksheets(1).Range(“A4″).Value = “Division Answer”**  
**Worksheets(1).Range(“B4″).Value = Answer**

Sửa đổi trên giúp chúng ta đưa thêm phép cộng kết hợp với phép nhân

**Answer = 8 + 8 / 4**

Và bạn cũng có thể kết hợp đủ thứ nhân chia cộng trừ và chỉ cần hiểu rằng nguyên tắc này hoàn toàn tương tự với nguyên tắc mà chúng ta đã được học. Excel VBA sẽ ưu tiên phép nhân, phép chia và nếu có nhân và chia thì phép toán được thực hiện từ trái qua phải. Sau khi tính toán xong nhân chia nó sẽ hỏi thăm đến cộng trừ và cũng thực hiện từ trái qua phải.

Đơn giản chỉ vậy thôi!

Vốn dĩ biến rất phức tạp nên trong phần tiếp theo, mình và bạn sẽ cùng thử tìm hiểu một vài dạng khác nhau của biến trong VBA nhé.

Tự học VBA – Bài 13: Những dạng biến khác nhau trong VBA

Ở các bài trước, mình vẫn thường nói về độ phức tạp của biến, bởi đơn giản một khi bạn đã nắm chắc cách sử dụng các biến thì đó chính là nền tảng cho việc code sau này. Sự phức tạp của biến nằm ở sự đa dạng về kiểu biến và cách sử dụng của nó. Chúng ta đã biết đến integer (Số nguyên) là một kiểu biến và trong bài này, mình và các bạn sẽ cùng tìm hiểu thêm một số kiểu biến nữa chẳng hạn:

**As Long**  
**As Single**  
**As Double**  
**As String**

Như trên thì chúng ta có **Long**, **Single**, **Double**, **Currency** lần lượt là các kiểu biến trong VBA. Sự khác nhau giữa chúng nằm ở khả năng lưu trữ và thể hiện ra các dấu thập phân mà bạn muốn ở các biến đó.

Đầu tiên là kiểu **Long** – Nó là sự thay thế tuyệt vời cho kiểu **Integer**, Sự thay thế tuyệt vời là sao nhỉ? Chẳng lẽ Integer còn có những khuyết điểm gì nữa? Đúng vậy, với kiểu Integer, chúng ta sẽ chỉ lưu giữa được các giá trị trong khoảng

-**32,768 =< integer =< 32,767 .**

(*Dấu phẩy phân cách nghìn chứ không phải thập phân đâu nhé các bạn ^\_^*)

Nếu bạn muốn lưu trữ một số lớn hơn thì rõ ràng 32,767 là không thể đủ (Giả sử đó là tiền Việt thì với 32,767 thì chỉ đủ ăn một bát phở thôi). Và đó chính là lý do mà kiểu**Long** được trọng dụng. Với Long bạn có thể lưu trữ biến trong khoảng giá trị lớn hơn đó là:

**-2, 147, 483, 648 =< long <= 2, 147, 483, 647**

Như vậy, cả Long và Integer đều lưu trữ các số nguyên, và đều có khoảng giá trị nhất định của nó. Bạn cũng không cần phải học thuộc khoảng giá trị này làm gì mà chỉ cần áng chừng là được. Số nguyên đã Ok nhưng đâu phải lúc nào chúng ta cũng chỉ thao tác với số nguyên đâu, trong công tác kế toán thì chẳng hiếm trường hợp chúng ta thực hiện chia 10 cho 3. Và khi đó chúng là lại cần phải có một kiểu biến khác.

Một số kiểu biến mà bạn có thể sử dụng để lưu trữ dạng biến có dấu thập phân (Số thực) đó là **Single** và **Double.**Với Single thì bạn có thể lưu trữ tới 4 bytes (Giá trị nằm trong đoạn – 10 mũ 37 đến 10 mũ 38). Còn với Double bạn có lưu được biến lên đến 8 bytes (Giá trị nằm ở mức mà bạn không cần phải nghĩ tới vì nó quá to so với tầm của mọi chương trình lập trình thông dụng). Nếu muốn chắc cú, chương trình nhẹ nhàng thì các bạn cứ xài Double hoặc không thì Single là đã đủ lắm rồi – *Tiết lộ cho các bạn một bí mất đó là hầu hết các biến trong*[***phần mềm kế toán excel***](http://www.h2account.com/)*H2 Account được đặt dưới dạng Double (Để tránh được mọi lỗi liên quan đến biến nếu có).*

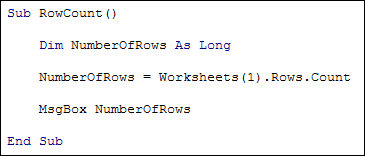
Chúng ta hãy thực hành luôn để hiểu về biến hơn với việc tạo một sub mới với tên RowCount và chèn vào giữa đoạn code sau:

**Dim NumberOfRows As Long**

**NumberOfRows = Worksheets(1).Rows.Count**

**MsgBox NumberOfRows**

Khi có, chương trình của bạn sẽ giống hình dưới đây:



Ta vừa thiết lập một biến với tên gọi NumberOfRows. Thay vì khai báo kiểu biến là Integer, ta sử dụng kiểu Long. Dòng thứ hai chúng ta gán biến đó bằng một giá trị của việc áp dụng phương thức Count (Đếm) cho đối tượng Worksheets(1) với thuộc tính Rows(số dòng). Còn dòng thứ 3 thì sao?

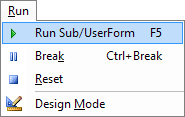
**MsgBox NumberOfRows**

MsgBox là viết tắt của từ Message Box tạm dịch là Hộp tin – Đại khái chúng ta sử dụng MsgBox để xuất dữ liệu ra ngoài màn hình. Và để có thể kiểm tra nhanh giá trị của biến NumberOfRows thì chúng tiến hành chạy thử chương trình. Hãy chắc chắn rằng con chuột đang nhấp nháy giữa cặp Sub và End Sub nhé!

Bây giờ, tại cái khu vực có cái nút mũi tên màu xanh ở phía trên thanh công cụ các bạn bấm click chuột trái.

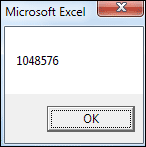


Đó là nút chạy chương trình, khi bạn click nó VBA sẽ thử chạy đoạn Sub của bạn. Một cách khác là bạn bấm vào dòng chữ Run trên thanh công cụ và click –> Run Sub/User Form.



Phím tắt để bạn chạy các chương trình của mình là phím F5.

Khi bạn chạy chương trình, bạn sẽ nhìn thấy một hộp thông báo xuất hiện

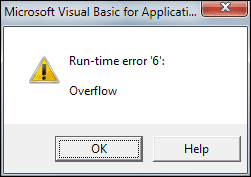


Số 1048576 chính là giá trị của biến NumberOfRows. Hay nói một cách khác là có hơn 1 triệu dòng trong phiên bản excel hiện tại (Excel 2013)

Click nút OK trên hộp thông báo của bạn và quay trở lại trình biên dịch. Bây giờ, bạn hãy thử thay thế kiểu biến **Long** bằng kiểu biến quen thuộc với chúng ta: **integer**

**Dim NumberOfRows As Integer**

Thử chạy lại Sub của bạn, khi đó, bạn sẽ nhìn thấy một thông báo lỗi sau:



Lỗi đó là Overflow (tạm dịch là tràn bộ nhớ). Bạn gặp lỗi này bởi vì giá trị bạn thử lưu trữ vào biến nó lớn hơn so với khả năng lưu trữ của biến đó. Như đã biết, kiểu Integer chỉ cho phép biến đó lưu trữ số nguyên lớn nhất là 32767. Trong khi với việc chạy lần đầu chúng ta đã biết rằng số dòng trong excel là hơn 1 triệu. Điều đó sẽ khiến chương trình báo lỗi như bạn đã thấy.

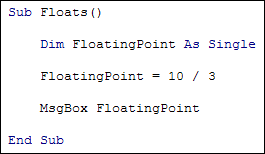
Để kiểm tra số thập phân, các bạn tạo một sub mới về tên gọi: Floats và thêm đoạn code dưới vào giữa cặp Sub – End Sub:

**Dim FloatingPoint As Single**

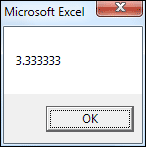
**FloatingPoint = 10 / 3**

**MsgBox FloatingPoint**

Đoạn code của chúng ta sẽ như hình dưới đây:



Đảm bảo con chuột đang nhấp nháy bên trong cặp Sub và End Sub –> Bấm F5 trên bàn phím để chạy chương trình. Bạn sẽ nhìn thấy một hộp thoại sau xuất hiện:



Các bạn có thể thấy một số thập phân được hiển thị trong hộp thông báo, nó trả lời cho phép chia 10 cho 3, sau dấu phẩy là thêm 6 chữ số nữa.

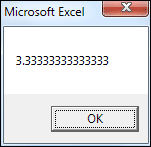
Đổi sang việc sử dụng kiểu biến **Double**

**Dim FloatingPoint As Single**

Sửa thành

**Dim FloatingPoint As Double.**

Chạy lại đoạn code và thông báo sau sẽ hiển thị:



Quan sát hộp thoại dễ dàng nhận thấy là biến FloatingPoint đã thể hiện giá trị chính xác hơn bằng việc xuất hiện thêm 14 chữ số đằng sau dấu phẩy.

Vì vậy, bạn có thể hình dung ra cách sử dụng **Single** và **Double** chưa? Nếu bạn muốn lưu trữ số liệu một cách chính xác nhất có thể thì bạn có thể sử dụng kiểu biến **Double**thay vì **Single**. Dưới đây là một ví dụ trực quan hơn:

Thay thế đoạn code ở giữa của bạn bằng đoạn code này:

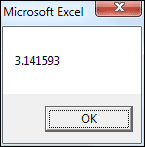
**Dim FloatingPoint As Single**

**FloatingPoint = WorksheetFunction.Pi**

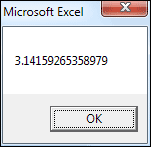
**MsgBox FloatingPoint**

Ở đây, ta thấy xuất hiện của  **WorksheetFunction** – là viết tắt của**Application.WorksheetFunction**, như vậy các bạn đã biết WorksheetFunction là gì chưa? Phương thức hay thuộc tính nhỉ? Nó là một thuộc tính của đối tượng Application. Đối tượng Application bạn hình dung nó chính là toàn bộ Excel của chúng ta, lớn hơn worksheet. Sau dấu chấm của WorksheetFunction các bạn sẽ thấy xuất hiện một danh sách các hàm excel mà ta có thể sử dụng, một trong số đó là Pi (số Pi)

Khi bạn chạy code, hộp thoại sau sẽ xuất hiện:



Tiếp thục làm thí nghiệm với việc thay thế kiểu biến **Single** thành **Double** –> Chạy code (F5) và quan sát hộp thoại sau:



Với kiểu biến Single, có 6 số đằng sau dấu chấm trong khi đó với kiểu biến Double chúng ta có đến 14 chữ số theo sau dấu chấm. Số cuối cùng của kiểu biến Single là 3 trong khi số thứ 6 (Sau dấu chấm) của kiểu biến Double là 2 bởi vì khi sử dụng Single, Excel đã tự làm tròn lên.

Vì vậy xin được nhắc lại: nếu bạn muốn sử dụng số chính xác hơn, giảm thiếu tối đa sự sai số thì biến kiểu biến Double là một lựa chọn không thể hợp lý hơn.

Thêm một loại biến nữa cũng được sử dụng đó là **Variant**

**Dim FloatingPoint As Variant**

Kiểu biến Variant được sử dụng khi bạn không chắc chắn về giá trị sẽ lưu trữ ở trong nó, có thể là số mà cũng có thể là chữ hoặc một đối tượng nào đó. Tuy nhiên, việc sử dụng biến này là hãn hữu bởi vì nếu bạn lạm dụng sẽ khiến chương trình của bạn chạy chậm. Biến có kiểu Variant sẽ chiếm 16 bytes bộ nhớ cho việc lưu trữ số và 22 bytes cho lưu trữ text.

Sẵn tiện đây, mình cũng nói luôn đó là có bạn sẽ thắc mắc: “**Tại sao phải làm ra nhiều kiểu biến rắc rối và phức tạp thế này làm gì nhỉ**? –> Xin được trả lời là khi chúng ta lập trình, chương trình của chúng ta không đơn giản chỉ là một cặp Sub và End Sub đơn giản mà nó có thể lên đến cả nghìn, chục nghìn cặp đó…Việc thiết lập đúng biến sẽ giúp bạn tiếp kiệm dung lượng bộ nhớ và cũng khiến cho phần mềm hoạt động nhanh hơn, linh hoạt và tránh bị treo.

**Kiểu biến chuỗi ký tự**

Nếu bạn muốn lưu trữ biến dưới dạng văn bản thì bạn có thể sử dụng kiểu biến String (Nhớ là String chứ không phải Sting đâu nhé ^\_^)

**Dim MyText As String**

Có rất nhiều các phương pháp xây dựng sẵn có mà bạn có thể sử dụng với chuỗi văn bản, chúng ta sẽ học kĩ ở các bài học sau, nắm vững kiểu biến này sẽ phát triển kỹ năng lập trình VBA của bạn rất nhiều đó.

 Sau đây mình có thể tổng kết và liệt kê một số kiểu biến thường dùng nhất trong Excel VBA

* **Byte** (Kiểu biến dữ liệu)
* **Bloolean** (Kiểu biến chỉ có 2 giá trị trả về Đúng (True) hoặc Sai (False)
* **Integer** (Kiểu biến số nguyên có giới hạn vùng giá trị -( nhỏ)
* **Long** (Kiểu biến số nguyên có giới hạn vùng giá trị – (lớn)
* **Currency** (Kiểu biến tiền tệ)
* **Single** (Kiểu biến số thực có giới hạn khả năng làm tròn 6 số, miền giá trị lớn )
* **Double** (Kiểu biến số thực có giới hạn khả năng làm tròn 14 số, miền gái trị cực lớn)
* **Date** (Kiểu biến ngày tháng)

Trong bài học tiếp theo, chúng ta sẽ tìm hiểu về LOGIC CÓ ĐIỀU KIỆN trong VBA

Tự học VBA – Bài 14: Câu lệnh IF

Những khái niệm nền tảng ở hai phần trước chắc hẳn phần nào đã giúp các bạn có hiểu biết cơ bản về VBA, một ngôn ngữ lập trình khá đơn giản được bác Bill đưa vào bộ office của mình. Thực tế đã chứng minh, nếu chúng ta học về VBA, hiểu về VBA và làm chủ được nó, các bạn sẽ cải thiện được năng suất làm việc của mình. Hành trình tự học VBA là rất gian nan, và thường sẽ gây ra nản. Bởi vậy, mà người thành công trong xã hội thường ít hơn nhóm người còn lại…

Khi bước sang phần này, tức là các bạn đã đang bước sâu vào thế giới lập trình khi mà mọi thứ được tạo ra đều đã trải qua sự tính toán của con người. Hay nói khác đi, mọi thứ xảy ra không phải do ngẫu nhiên mà là do chúng ta sắp đặt. Để làm được điều này, các nhà lập trình sử dụng đến các câu lệnh mang tính điều kiện. Tức là để một sự kiện B xảy ra thì nó phải đảm bảo được một điều kiện A nào đó. Và khi điều kiện A nào đó không có thì điều gì sẽ xảy ra…? Đương nhiên sự kiện B sẽ không xuất hiện đúng không nào. Nhưng sự kiện C sẽ có cơ hội nhen nhóm sinh ra…

Mình có thể nói vui như thế này: Hãy giả sử có ba con: Sư tử – Linh cẩu – Thỏ .  Để sự kiện B (Linh cẩu chén Thỏ)  xảy ra thì phải đảm bảo được điều kiện là không có Sư tử. Nhưng nếu có sư tử thì sự kiện C ( Sư tử chén Linh cẩu) sẽ có cơ hội nhen nhóm sinh ra… =)))

Mọi sự nếu nếu, thì thì đó trong lập trình VBA và trong hầu hết các ngôn ngữ lập trình khác thì người ta đều dùng câu lệnh IF.   với kiểu luận tiếng Việt là:  Nếu…. thì….

**Câu lệnh IF**

Chúng ta  hãy làm quen với một vài câu nếu rất thú vị được lấy từ cuộc sống thường ngày dưới đây:

***IF (nếu) tôi có đôi giày kia thì tôi sẽ rất hạnh phúc***

***IF (nếu) tôi ăn nhiều kẹo thì tôi sẽ bị sún răng***

***IF (nếu) tôi truy cập h2account.com thì tôi sẽ nhận được các bài học VBA miễn phí***

Dễ nhận thấy là khi lập trình, chúng ta sẽ phải phụ thuộc rất nhiều bởi câu lệnh IF này bởi lập trình là phải lập ra một chương trình có thể chạy khi tiếp nhận các kết quả khác nhau. Mỗi kết quả khác nhau lại dẫn chúng ta đến với một công việc khác nhau và khi đó IF phát huy hết hiệu quả của mình.

Cấu trúc câu lệnh IF trong VBA sẽ giống như thế này:

**If Condition\_To\_Test Then**

**‘CODE HERE**

**End If**

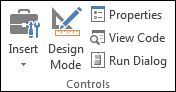
Bây giờ hãy cùng mình phân tích kĩ câu lệnh này và các bạn buộc phải ghi nhớ, nằm lòng nó một cách gọi là “Khắc cốt ghi tâm” bởi nếu quên các bạn sẽ không bao giờ trở thành một lập trình viên được. Trước tiên, bạn bắt đầu với từ khóa **If** (Chữ cái I viết hoa và chữ cái f viết thường. Cho dù bạn viết hoa hết hoặc viết thường hết thì excel sẽ tự động chuyển về dạng chuẩn này) sau đó một khoảng cách là điều kiện mà bạn muốn kiểm tra. Điều kiện này có thể đúng (TRUE) hoặc sai (FALSE). Theo sau điều kiện là từ khóa **Then** (Viết hoa chữ  T) . Và câu lệnh IF được đóng lại với cụm từ khóa **End if**

Ở giữa If và End If là đoạn code của bạn nhưng đoạn code này chỉ được thực hiện nếu điều kiện mà bạn muốn kiểm tra là đúng (TRUE). Ngược lại nếu điều kiện lại là sai (FALSE) thì VBA sẽ bỏ qua phần này và chạy tới End If rồi tiếp tục tiến trình khác đã được lập trình (nếu có)

\* *Có một lưu ý nhỏ là đoạn code ở giữa có thể có hoặc không nhưng cụm từ khóa màu đỏ và biểu thức điều kiện cần kiểm tra là bắt buộc phải có nếu không VBA sẽ báo lỗi ngay. Và không giống như việc enter khi tạo sub, các bạn phải tự đánh cụm từ khóa End If khi muốn kết thúc câu lệnh If của mình.*

Cách tốt nhất để ghi nhớ là thực hành nhiều lần, các bạn hãy thả lỏng đầu óc, thư giãn và thật thoải mái nhé. Chúng ta sẽ tạo một file excel mới cho chuỗi bài học về các phép toán LOGIC có điều kiện và đừng quên lưu nó dưới dạng **Excel Macro- Enable Workbook, và file excel đó sẽ có đuổi XLSM và có dấu chấm than to đùng màu vàng đó**

Tiếp theo, các bạn click vào tab Developer (Ví Thạch Sanh dành cho các bạn nào còn nhớ) sau đó bấm vào biểu tượng VIew Code trong mục Controls.



Việc click View Code sẽ mở  ra trình biên dịch VBA . Và nếu như vì lý do nào đó cửa sổ viết code không hiện ra thì các bạn có thể click đúp vào sheet1 bên tay trái trình biên dịch VBA. Và nếu như các bạn lại không nhìn thấy cái Sheet1 nào cả thì hãy chắc chắn dùm mình là đây là file excel mới, nếu đã chắc chắn điều này thì các bạn bấm vào View –> Project Explorer là xong

Tạo một sub mới trong cửa sổ viết code với tên gọi: If\_Test\_1. và chèn đoạn code dưới đây vào giữa

**Dim MyNummber As Integer**

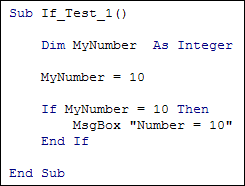
**MyNumber = 10**

**If MyNumber = 10 Then**

**MsgBox “Number = 10″**

**End If**

Đoạn code của bạn sẽ trông giống như thế này:



Ở đoạn code giữa, chúng ta thiết lập một biến nguyên được gọi là MyNumber. Tiếp theo gán giá trị cho nó nó bằng số 10 thông qua dấu bằng. Các bạn nếu không thích số 10 thì có thể lưu trữ là một số tùy chọn như ngày sinh của mình chẳng hạn, nhưng lưu ý là nếu thay thì thay đồng nhất cả các dòng bên dưới nhé. Giờ chúng ta bàn sâu đến nhân vật chính của ngày hôm nay. Câu lệnh If.

**If MyNumber = 10 Then**

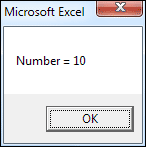
Bạn có từ khóa “If”” và từ  khóa “Then”, giữa hai từ khóa chúng ta có điều kiện mà chúng ta muốn kiếm tra. Điều đó là:

**MyNumber = 10**

Nhìn thì rất giống nhau giữa việc gán giá trị bên trên và điều kiện mà chúng ta muốn kiểm tra, nhưng ý nghĩa của nó hoàn toàn khác nhau nhé. Nếu như ở bên trên chúng ta gán giá trị cho MyNumber là 10. thì ở bên dưới, chúng ta đưa việc MyNumber = 10 là điều kiện để kiểm tra tính chính xác.

Bằng cách diễn đạt rằng: Nếu MyNumber có giá trị bằng 10, bạn đang tạo ra hai khả năng TRUE (Đúng) hoặc FALSE (Sai). Và đó là những gì mà VBA sẽ kiểm tra. (*Chẳng riêng gì VBA đâu, mọi ngôn ngữ lập trình đều kiểm tra theo kiểu TRUE, FALSE này. À, mà hình như trong excel câu lệnh IF đơn giản cũng kiểm tra như vậy mà ^^*)

Chẳng cần nói nhiều, thì ai cũng biết rằng điều kiện mà chúng ta đặt ra ở đây chắc chắn là đúng rồi, các bạn đặt dấu nhấp chuột vào đoạn code giữa Sub và End Sub, sau đó ấn F5 để chạy chương trình. Bạn sẽ nhìn thấy hộp thoại hiển thị giống như hình dưới đây:



Click OK trên hộp thông báo để quay trở lại cửa sổ code. Bây giờ, thử thay đổi việc gán giá trị biến MyNumber từ 10 thành 11. Cũng lại đặt con chuột dư trên và ấn nút F5 lại lần nữa để chạy code. Ầu, chẳng có cái gì hiển thị ra ở đây nữa cả.

Lý do dẫn đến việc không có hiện tượng xảy ra nằm ở chỗ: Trong câu lệnh IF mà các bạn đã viết ở trên, chúng ta chỉ ra điều kiện cho chương trình nếu điều kiện được đưa vào kiểm tra xảy ra giá trị TRUE (đúng). Tức là nếu MyNumber có giá trị bằng 10 thì sẽ hiển thị hộp thoại thông báo có nội dung “Number = 10″.  Bởi vậy, mà khi gán giá trị biến MyNumber = 11 tức là điều kiện đã FALSE (sai) và ứng dụng sau khi ấn F5 không có hiện tượng gì xảy ra. VBA đã bỏ qua khi câu lệnh FALSE.

Trong phần tiếp theo, chúng ta sẽ học cách điều khiển VBA trong câu lệnh IF để ngay cả khi FALSE xảy ra chúng ta vẫn luôn kiểm soát được tình hình bằng cách sử dụng **Else** và**Elseif**