# LẬP TRÌNH VISUAL BASIC
# BÀI TẬP THỰC HÀNH CHƯƠNG 5 (TIẾP THEO)

## CHƯƠNG TRÌNH CON VÀ DEBUG CHƯƠNG TRÌNH

## A./ Chuẩn đầu ra

Sau khi học xong chương này, sinh viên có thể:

### Về kiến thức

Trình bày được các hàm Visual Basic cung cấp sẵn. Áp dụng trong các yêu cầu cụ thể.

Phân biệt sự khác biệt giữa thủ tục xử lý sự kiện, thủ tục do người dùng định nghĩa.

Trình bày được các định nghĩa thủ tục (Sub), hàm (Function).

Trình bày được các khái niệm:

- Tham số hình thức: Tham trị, tham biến.
- Tham số thực.
- Truyền tham số.
- Biến toàn cục, biến cục bộ, tầm tác động của biến.

Chạy tay từng lệnh một chương trình VB đơn giản có sử dụng chương trình con và cho biết giá trị các biến sau mỗi câu lệnh.

### Về kỹ năng

Xây dựng thành thạo các loại chương trình con trong Visual Basic.

Hình thành kỹ năng gỡ rối chương trình.

## B./ Chuẩn bị

SV phải làm các bài tập đã làm giờ lý thuyết và các bài tập phần thực hành.

## C/. Phương tiện

Phòng máy có cài chương trình Microsoft Visual Basic 6.0

## D./ Lý thuyết

Trong VB, đơn vị chức năng nhỏ nhất mà người lập trình có thể xây dựng và dùng (gọi) lại nhiều lần trong chương trình là chương trình con.

Có 2 loại chương trình con là hàm (Function) và thủ tục (Sub)

### I./ Hàm (Function)

Hàm gồm một tập các dòng lệnh dùng để thực hiện một tác vụ cụ thể. Hàm luôn luôn trả về một giá trị thông qua tên hàm.

Visual Basic cung cấp nhiều hàm có sẵn cho phép phát triển ứng dụng một cách nhanh chóng, dễ dàng. Ví dụ:

| Hàm | Ý nghĩa | Ví dụ |
|---|---|---|
| Sqr(x) | Căn bậc 2 | Sqr(4) trả về 2 |
| Round(x[,n]) | Làm tròn đến n chữ số phần thập phân | Round(34.673,2) trả về 34.67 |
| Asc(c) | Mã Ascii của ký tự c | Asc('a') trả về 97, asc('A') trả về 65 |
| Chr(n) | Ký tự có mã n | Chr(65) trả về ký tự A |
| Val(s) | Đổi chuỗi thành số | Val("1234") trả về số 1234 |
| Str(n) | Đổi số thành chuỗi | Str(12.45) trả về chuỗi " 12.45" |
| Ucase(s) | Đổi chuỗi thành chữ in | Ucase("aBCd") trả về chuỗi "ABCD" |
| Lcase(s) | Đổi chuỗi thành chữ thường | Ucase("aBCd") trả về chuỗi "abcd" |

Có thể tự tạo cho mình các hàm để thực hiện công việc nhất định và có thể dùng lại về sau.

## 1) Định nghĩa hàm

**[Private/Public] Function <Tên hàm>([<Danh sách tham số>]) [As <Kiểu của hàm>]**
[<Khai báo>]
<Lệnh>
<Tên hàm = giá trị>
**End Function**

Định nghĩa hàm gồm 3 phần chính: Tiêu đề, thân hàm, kết thúc.

### a) Tiêu đề

Tiêu đề là dòng đầu tiên trong định nghĩa hàm, trong đó:

*Public, Private* qui định tầm vực của hàm. Hàm được định nghĩa với từ khoá Private chỉ được sử dụng trong form chứa nó, hàm được định nghĩa với từ khoá Public có thể sử dụng trong các form khác.

*<Tên hàm>* do người dùng đặt, dùng để gọi hàm sau này.

*<Danh sách tham số>* là danh sách các tham số hình thức (Gọi là tham số hình thức vì khi định nghĩa hàm các tham số này chưa có nội dung). Danh sách tham số hình thức có dạng:

[*ByVal*|*ByRef*] <Tên tham số> [()] [As <Kiểu của tham số>]

*ByVal* qui định tham số hình thức nhận giá trị từ tham số thực (là tham số cung cấp lúc gọi hàm). Nói cách khác, tham số thực có thể là biến, hằng, biểu thức, hàm.

*ByRef* qui định tham số hình thức nhận địa chỉ từ tham số thực. Tham số thực phải là biến.

Có thể có nhiều tham số hình thức, các tham số hình thức cách nhau bởi dấu phẩy.

*<Kiểu của hàm>* là kiểu dữ liệu trả về khi dùng hàm

### b) Thân hàm

Thân hàm chứa lệnh thực hiện các hành động.

Thao tác trên tham số hình thức của danh sách tham số.

Các tham số hình thức chỉ được dùng trong thân hàm của nó.

Ra khỏi hàm, tên của hàm phải nhận một giá trị thuộc kiểu đã khai báo.

### c) Kết thúc

Kết thúc phần định nghĩa hàm bằng từ khóa ***End Function***

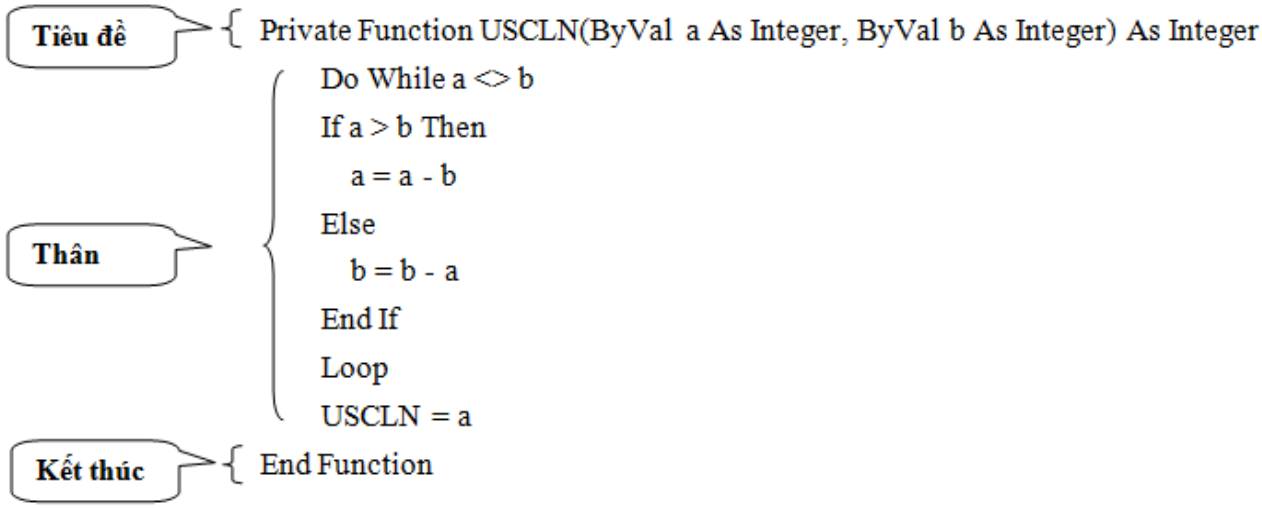
### d) Dùng hàm

Lời gọi hàm phải là thành phần của một biểu thức.

Cú pháp gọi hàm thực thi: *<Tên hàm>([<danh sách tham số thực>])*.

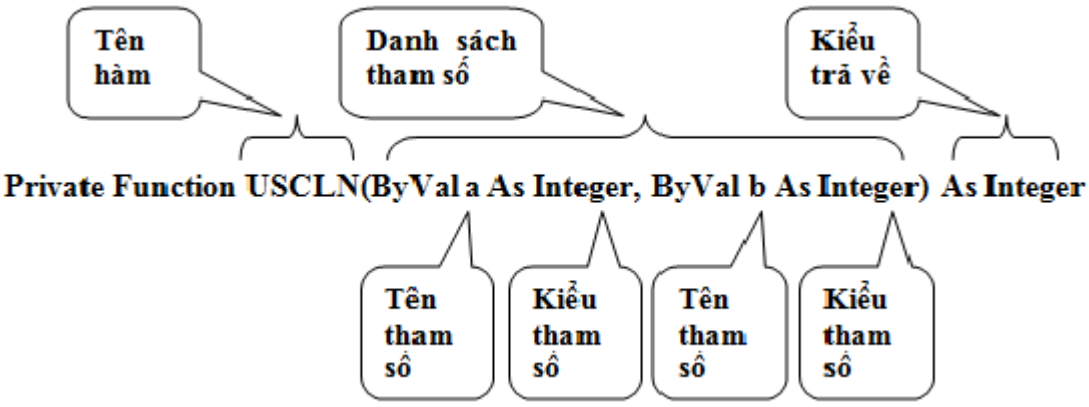
## 2) Ví dụ

Định nghĩa hàm *USCLN* tính ước số chung lớn nhất của hai số a, b.



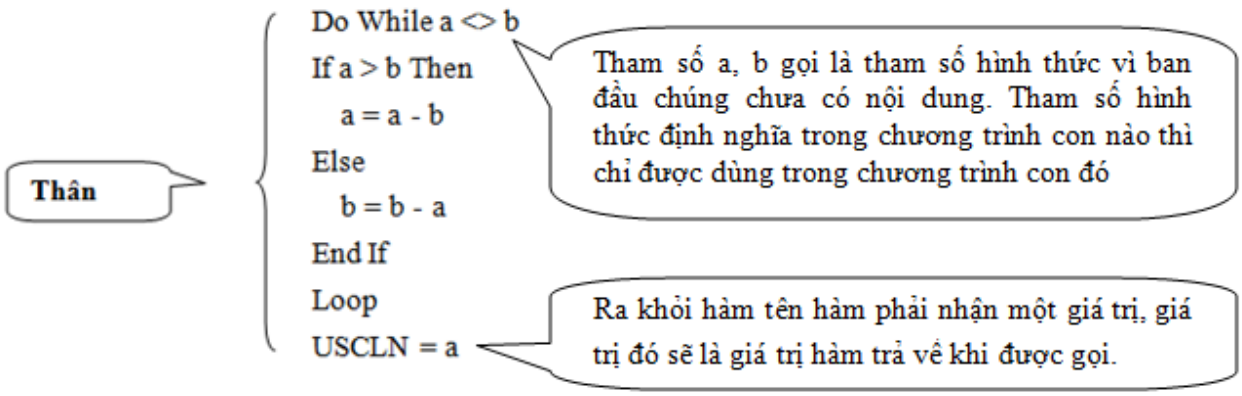

---

### a) Tiêu đề:




---

### b) Thân hàm:




### c) Khi dùng hàm có thể gọi:

```
MsgBox USCLN(24, 32)
```

Hoặc

```
x = 24
y = 32
Text1.Text = USCLN(x, y)
```

### 3) Hàm đệ qui

Hàm tự gọi lại chính nó trong khi thực hiện các lệnh gọi là hàm đệ qui (recursive).

Hàm đệ qui thường rất gọn nhưng khi thi hành sẽ sử dụng bộ nhớ rất nhiều do sinh ra biến cục bộ liên tục.

Đa số các bài toán có dạng $f_n(x) = g(f_{n-1}(x))$ đều có thể viết thành hàm đệ qui.

Ví dụ sau đây định nghĩa hàm tính n! theo giải thuật vòng lặp và giải thuật đệ qui.

```
----------------------------------------------------------------------
Private Function Giaithua(ByVal n As Integer) As Long
    Dim gt As Long
    If n <= 0 Then ' neu n<=0 thi ham tra ve -1
       Giaithua = -1
    Exit Function
    End If
    gt = 1
    Do While n > 1
       gt = gt * n
       n = n - 1
    Loop
    Giaithua = gt
End Function
----------------------------------------------------------------------------
Private Function Giaithua_Dequi(ByVal n As Integer) As Long
    Dim gt As Long
    If n <= 0 Then ' neu n<=0 thi ham tra ve -1
```

```
    Giaithua_Dequi = -1
  Exit Function
  End If
  If n > 1 Then
    Giaithua_Dequi = n * Giaithua_Dequi(n - 1)
  Else
    Giaithua_Dequi = 1
  End If
End Function
```

---

## II./ Thủ tục (Sub)

Thủ tục gồm một tập các dòng lệnh dùng để thực hiện một tác vụ cụ thể khi được gọi. Thủ tục không trả về giá trị nào. Visual Basic cung cấp nhiều thủ tục có sẵn cho phép phát triển ứng dụng một cách nhanh chóng, dễ dàng. Có thể tự tạo cho mình các thủ tục để thực hiện công việc nhất định và có thể dùng lại về sau.

Có 2 loại thủ tục là thủ tục tổng quát (General procedure) và thủ tục xử lý sự kiện (Event procedure).

- Thủ tục tổng quát được kích hoạt bằng lệnh gọi trong chương trình.
- Thủ tục xử lý sự kiện được kích hoạt khi có một sự kiện tác động lên form hoặc đối tượng điều khiển trên form. Thủ tục xử lý sự kiện thường có tên là <tên đối tượng>_<tên sự kiện>. Ví dụ Form_Load hoặc Commad1_Click...

### 1) Định nghĩa thủ tục

**[Private/Public] Sub <Tên thủ tục>([<Danh sách tham số>])**
[<Khai báo>]
<Lệnh>
**End Sub**

Định nghĩa thủ tục cũng gồm 3 phần chính: Tiêu đề, thân thủ tục, kết thúc.

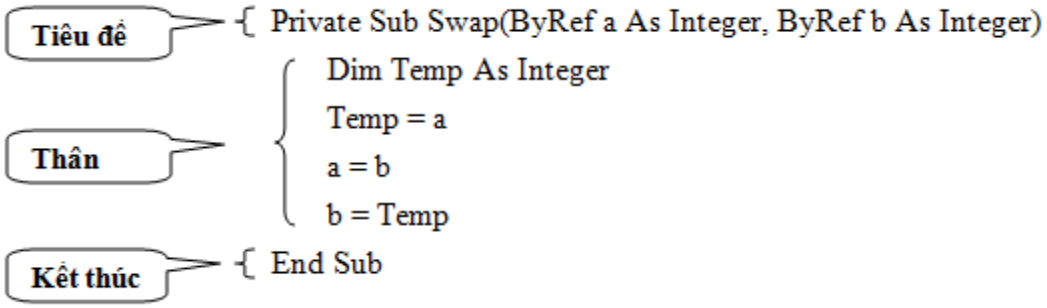
Các từ khóa *Public, Private* qui định tầm vực, *<Tên thủ tục>* , *<Danh sách tham số>* giống phần định nghĩa hàm.

Để gọi thủ tục thực thi, ta có 2 cách:

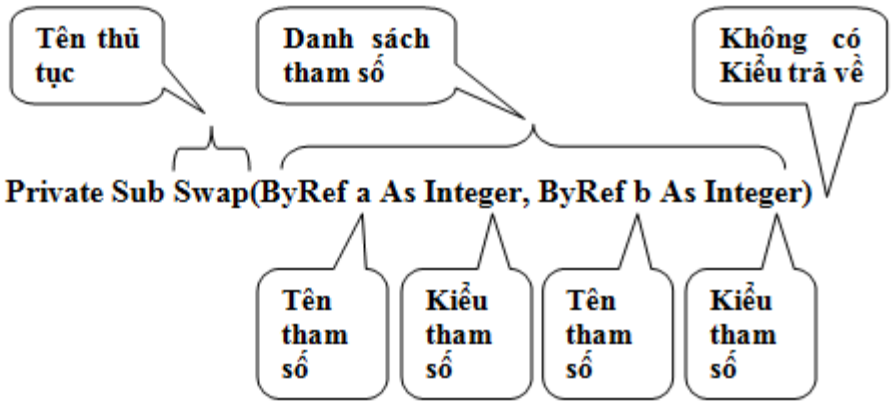
- <Tên thủ tục> [<Các tham số thực tế>]
- **Call** <Tên thủ tục> ([<Các tham số thực tế>])

### 2) Ví dụ

Định nghĩa thủ tục *Swap* hoán đổi giá trị hai biến a, b.




**a) Tiêu đề**




**b) Khi dùng thủ tục có thể gọi:**

```
x = 24
y = 32
Swap x, y
‘ hoặc
Call Swap(x, y)
```

## III./ Cơ chế truyền tham số

### 1) Khái niệm

Tham số hình thức: là tham số khi định nghĩa hàm, thủ tục.

Tham số thực: là tham số khi gọi hàm, thủ tục.

Nguyên tắc truyền tham số: Số lượng bằng nhau, phù hợp về kiểu dữ liệu.

Khi chương trình con được gọi, tham số thực sẽ được gán vào thay thế tham số hình thức tương ứng.

Có hai cách để truyền tham số cho chương trình con: Truyền bằng giá trị & truyền bằng địa chỉ.

**Truyền tham số bằng giá trị:**

Với cách truyền tham số theo cách này, mỗi khi một tham số được truyền vào, một bản sao của biến đó được tạo ra. Nếu chương trình con có thay đổi giá trị, những thay đổi này chỉ tác động lên bản sao của biến. Trong VB, từ khóa *ByVal* được dùng để xác định tham số được truyền bằng giá trị.

**Truyền tham số bằng địa chỉ:**

Truyền tham số theo địa chỉ cho phép chương trình con truy cập vào giá trị gốc của biến trong bộ nhớ. Vì thế, giá trị của biến có thể sẽ bị thay đổi bởi đoạn mã lệnh trong chương trình con. Mặc nhiên, trong VB6 các tham số được truyền theo địa chỉ; tuy nhiên ta có thể chỉ định một cách tường minh nhờ vào từ khóa *ByRef*.

Cách gán tham số thực thay đổi tùy theo *ByVal* hay *ByRef*:

*ByVal*: Tham số thực có thể là biến, hằng, biểu thức, hàm.

*ByRef*: Tham số thực phải là biến.

## 2) Ví dụ

### a) Ví dụ 1

Minh họa cách dùng các từ khóa *ByVal, ByRef* cho thủ tục hoán đổi giá trị các tham số a, b.

---

```
Private Sub Swap(ByRef a As Integer, ByRef b As Integer)
    Dim Temp As Integer
    Temp = a
    a = b
    b = Temp
End Sub
```

---

```
Private Sub Swap_01(a As Integer, b As Integer)
    Dim Temp As Integer
    Temp = a
    a = b
    b = Temp
End Sub
```

---

```
Private Sub Swap_02(ByVal a As Integer, ByVal b As Integer)
    Dim Temp As Integer
    Temp = a
    a = b
    b = Temp
End Sub
```

---

Khi dùng các thủ tục trên được kết quả như sau:

```
    …
    …
    Dim x As Integer, y As Integer
    x = 5
```

```
y = 8
Call Swap(x, y)              ‘ x=8, y=5
Call Swap_01(x, y)           ‘ x=8, y=5
Call Swap_02(x, y)           ‘ x=5, y=8
…
…
```

Với mục đích chuyển đổi giá trị hai biến x, y sau khi gọi thủ tục thì thủ tục Swap_02 không đạt yêu cầu, nguyên nhân là do ta dùng từ khóa *ByVal* trong danh sách tham số.

### b) Ví dụ 2

Minh họa cách dùng các từ khóa *ByVal, ByRef* cho hàm tính ước số chung lớn nhất của hai số a, b.

```
----------------------------------------------------------------------------------------
Private Function USCLN(ByVal a As Integer, ByVal b As Integer) As Integer
    Do While a <> b
    If a > b Then
       a = a - b
    Else
       b = b - a
    End If
    Loop
    USCLN = a
End Function
----------------------------------------------------------------------------------
Private Function USCLN_01(a As Integer, b As Integer) As Integer
    Do While a <> b
    If a > b Then
       a = a - b
    Else
       b = b - a
    End If
    Loop
    USCLN = a
End Function
----------------------------------------------------------------------------------
```

Khi dùng các hàm trên được kết quả như sau:

```
…
…
Dim x As Integer, y As Integer
x = 24
```

y = 32
Msgbox USCLN(x, y) & Space(5) & x & Space(5) & y ‘ In ra 8 24 32
Msgbox USCLN_01(x, y) & Space(5) & x & Space(5) & y ‘ In ra 8 8 8
…
…

Với mục đích tính ước số chung lớn nhất của hai biến x, y nhưng không làm thay đổi giá trị của x, y sau khi gọi hàm thì hàm USCLN_01 không đạt yêu cầu, nguyên nhân là do ta không dùng từ khóa *ByVal* trong danh sách tham số.

### 3) Chú ý

Qua ví dụ truyền tham số trong thủ tục Swap và hàm USCLN, một câu hỏi đặt ra là khi định nghĩa hàm, thủ tục, với yêu cầu nào thì dùng từ khóa ByVal, ByRef cho đúng?

Với các tham số đóng vai trò cung cấp tư liệu đầu vào cho hàm, thủ tục thì ta nên dùng từ khóa ByVal. Còn khi muốn lưu giữa sự thay đổi của tham số thì ta chọn từ khóa ByRef.

## IV./ Biến toàn cục, biến cục bộ, tầm tác động của biến

Biến toàn cục: là biến được khai báo “bên ngoài” tất cả các hàm\thủ tục.

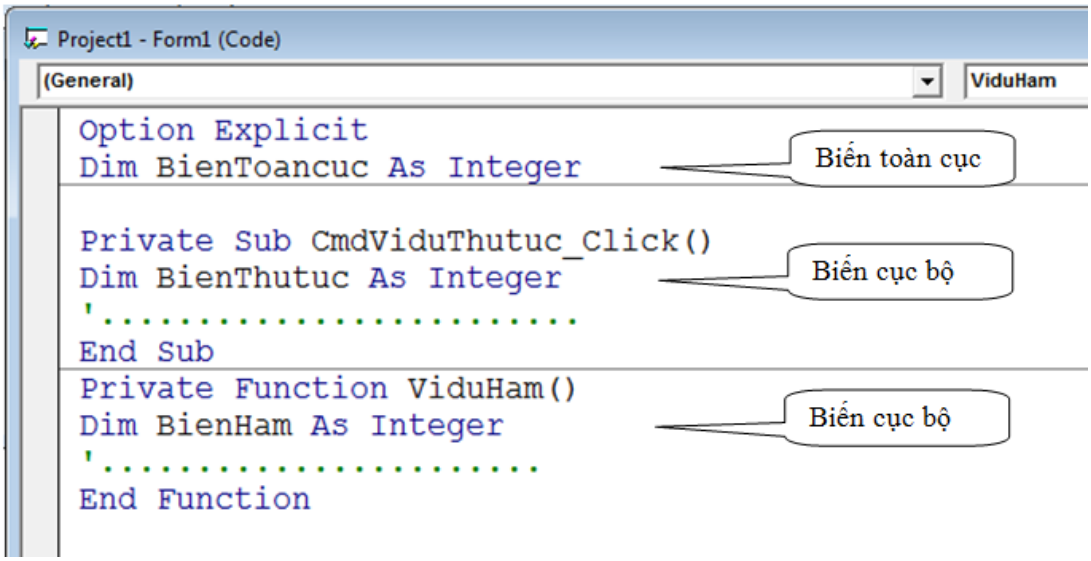
Biến cục bộ: là biến được khai báo “bên trong” hàm\thủ tục.

Khi khai báo biến toàn cục, tất cả các hàm\thủ tục đều có thể truy cập tới biến. Biến toàn cục có thời gian sống cùng với thời gian chạy chương trình.

Biến cục bộ có phạm vi từ vị trí khai báo đến cuối hàm\thủ tục. Biến cục bộ có thời gian sống từ lúc hàm\thủ tục được thực thi đến lúc kết thúc hàm\thủ tục.

Sau đây chúng ta giới hạn khái niệm khai báo biến toàn cục trong cửa số code của một form, khi đó biến toàn cục được khai báo trong phần đầu General

Hình dưới đây sẽ cho ta thấy vị trí của khai báo biến:

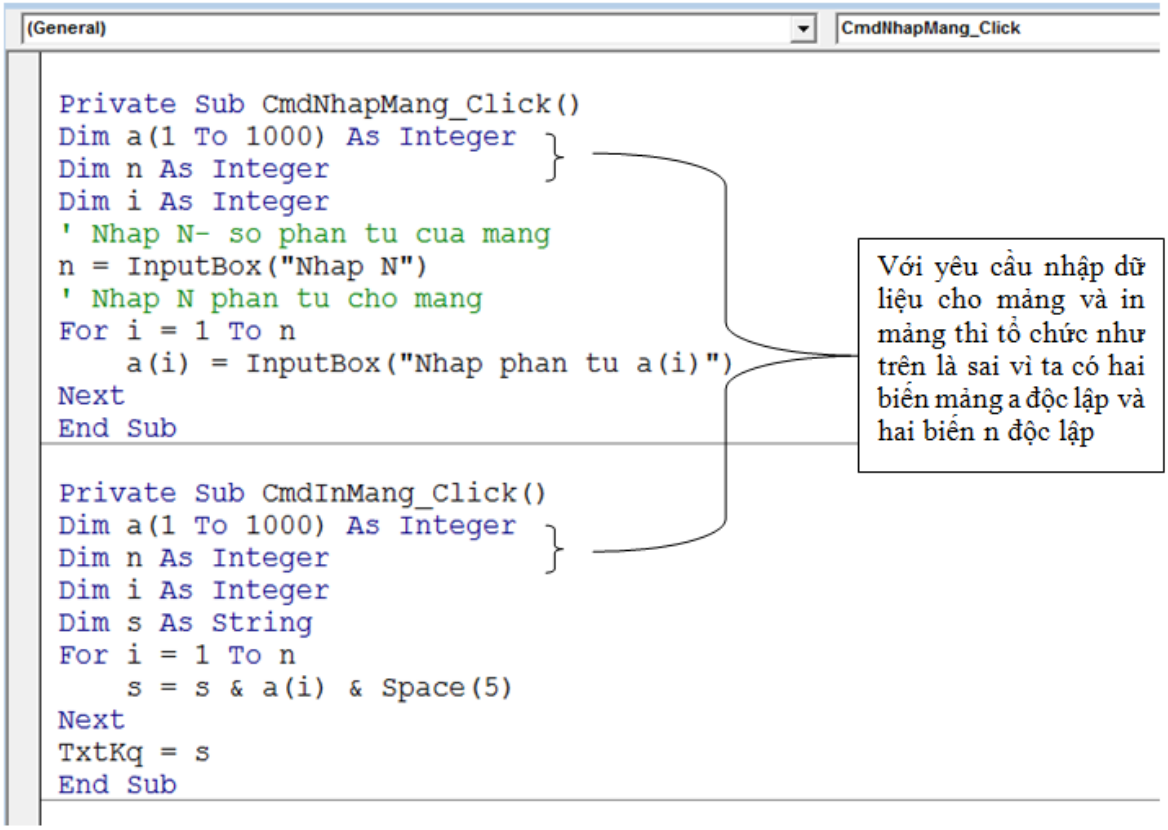



### Lưu ý 1:

Trong một hàm\thủ tục, ta có thể khai báo biến trùng tên với biến ở hàm\thủ tục khác. Tên biến khi này không phải là một biến duy nhất mà là các biến khác nhau với tác dụng khác nhau. Ví dụ

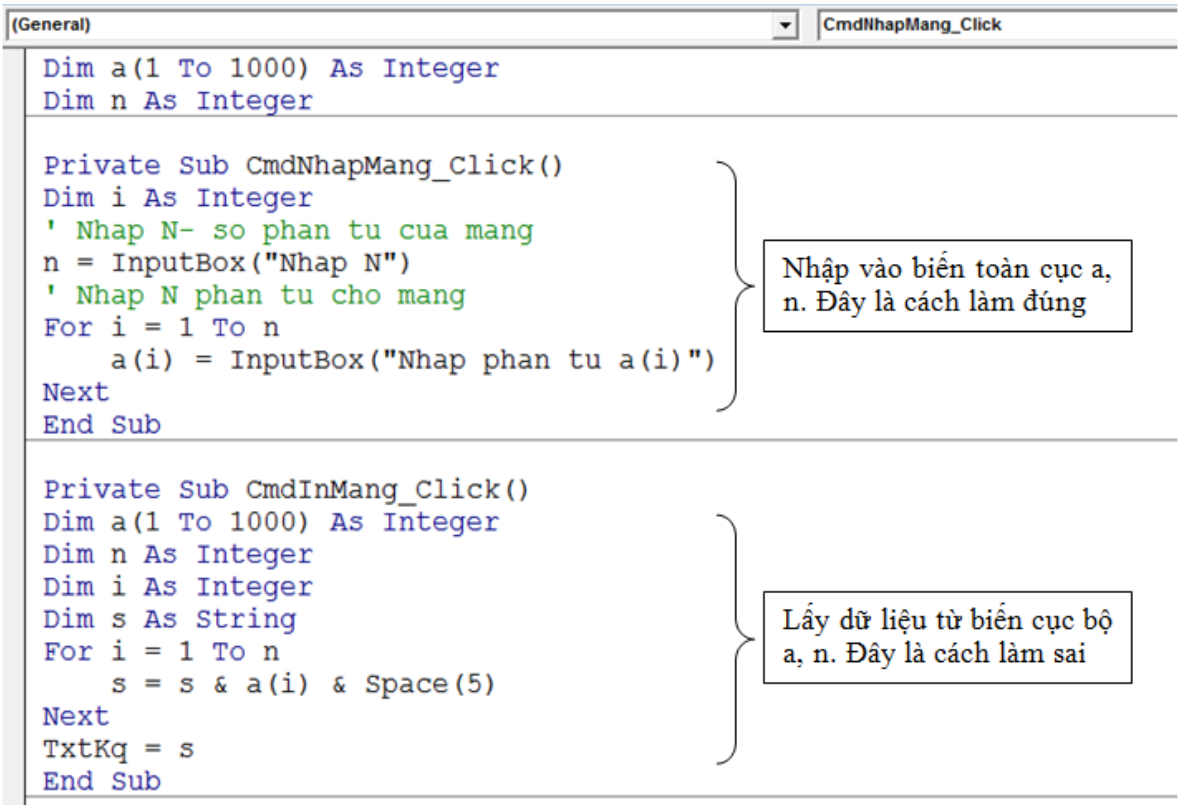
trong hàm\thủ tục A có biến cục bộ X và trong hàm\thủ tục  B có biến cục bộ cũng tên là X. Lúc đó, máy sẽ dùng hai ô nhớ khác nhau để lưu trữ hai biến, khi ra khỏi hàm\thủ tục A, biến cục bộ X trong hàm\thủ tục A mất tác dụng và khi ra khỏi hàm\thủ tục B, biến cục bộ X trong B mất tác dụng.

Ví dụ sau minh họa sai lầm trong tổ chức dữ liệu:




**Lưu ý 2:**

Biến cục bộ "che phủ" biến toàn cục nghĩa là nếu có tên biến cục bộ trùng tên biến toàn cục thì khi ở trong hàm\thủ tục có biến trùng tên, chương trình ưu tiên dùng tên biến cục bộ,. Khi vào các hàm\thủ tục khác, chương trình dùng tên biến toàn cục.




## V./ Kỹ năng gỡ rối chương trình (debug)

Gỡ rối chương trình (debug) là kỹ năng cơ bản nhất đối với mỗi lập trình viên. Tuy nhiên, kỹ năng này của sinh viên là rất yếu. Chương trình lập ra có thể hoạt động không theo ý muốn của người lập trình như: không hoạt động, có lỗi khi chạy, hoạt động nhưng không cho kết quả đúng, …

Khi chương trình hoạt động không theo dự kiến, cần phải khoanh vùng xem lỗi có thể phát sinh từ phần nào của chương trình. Sau đó, sử dụng các công cụ gỡ rối để tìm hiểu hoạt động của chương trình tại vị trí đó để xác định cụ thể xem lỗi bắt nguồn từ đâu. Sau đây là những kiến thức và thao tác cơ bản nhất để gỡ rối chương trình trong VB.

### 1) Sử dụng MsgBox

MsgBox được sử dụng để hiển thị các giá trị trong khi chương trình đang hoạt động. Sau đây chúng ta cùng xem một ví dụ:

```
Private Sub Command1_Click()
    Dim a As Integer
    a = 10
    a = a * a
    a = a * a
    a = a * a
    MsgBox a
End Sub
```

Đoạn chương trình trên khi chạy sẽ báo lỗi. Để biết được nguyên nhân, ta sửa lại chương trình như sau:

```
Private Sub Command1_Click()
    Dim a As Integer
    a = 10
    a = a * a
    MsgBox a
    a = a * a
    MsgBox a
    a = a * a
    MsgBox a
End Sub
```

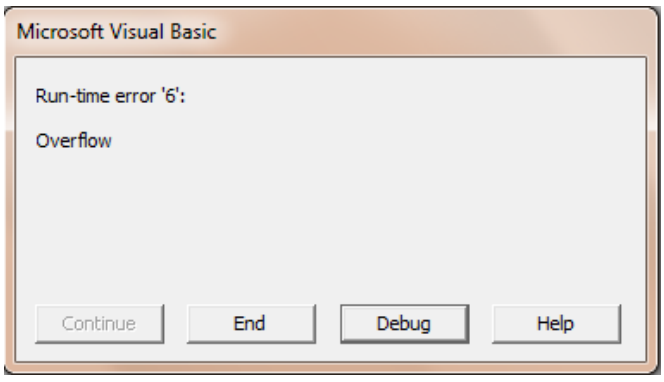
Khi chạy đoạn chương trình này, chúng ta sẽ thấy hộp thoại lần lượt hiện ra với các giá trị 100, 100000. Click OK tiếp, chương trình báo lỗi, có nghĩa là lệnh 10000*10000 không được thực hiện. Đến đây chúng ta có thể biết được lỗi này xuất hiện do giá trị 10000*10000 vượt quá khả năng lưu trữ của số nguyên Integer. Để chương trình hoạt động đúng dự kiến, ta chỉ cần sửa Integer thành Long.

### 2) Sử dụng cửa sổ Watch

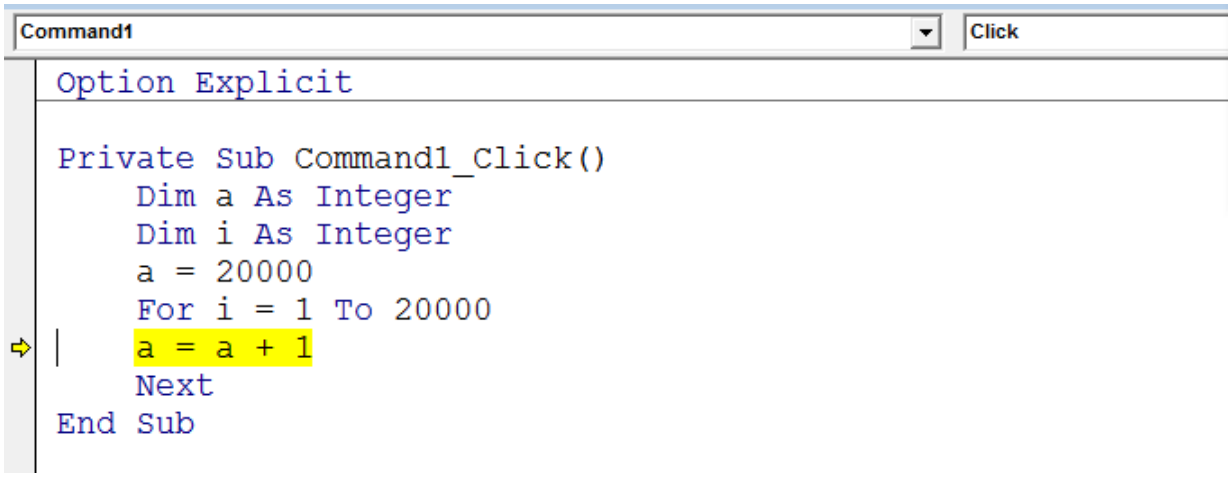
Xét ví dụ sau:

```
Private Sub Command1_Click()
    Dim a As Integer
    Dim i As Integer
    a = 20000
    For i = 1 To 20000
            a = a + 1
    Next
End Sub
```

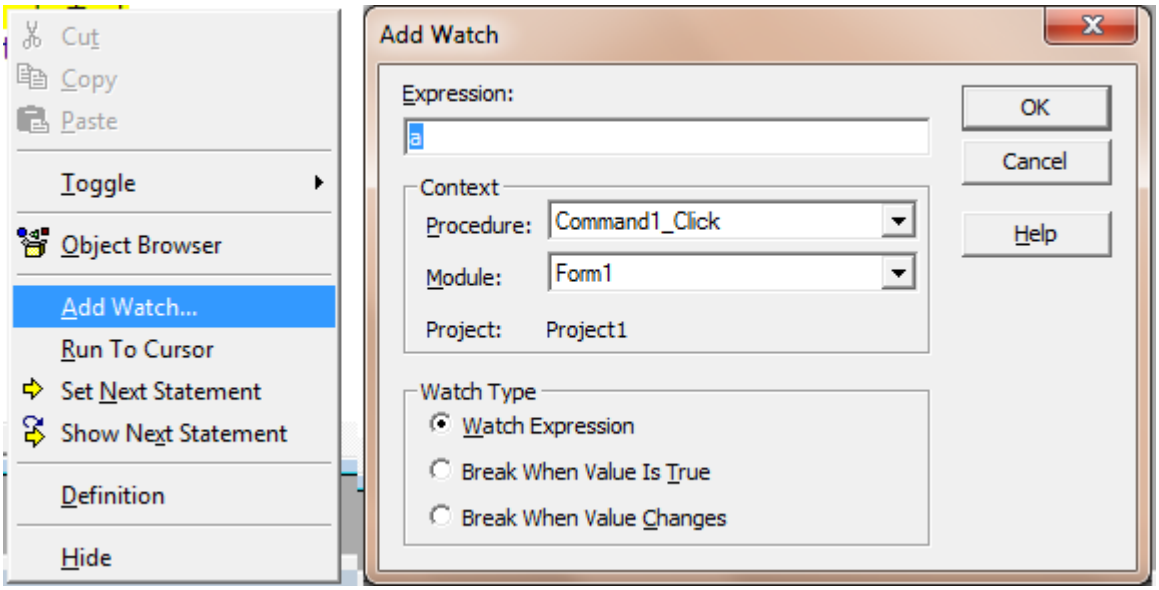
Đoạn chương trình này cũng phát sinh lỗi, tuy nhiên không thể sử dụng MsgBox để hiển thị kết quả trong từng trường hợp. Trong trường hợp này, khi chương trình báo lỗi như sau:



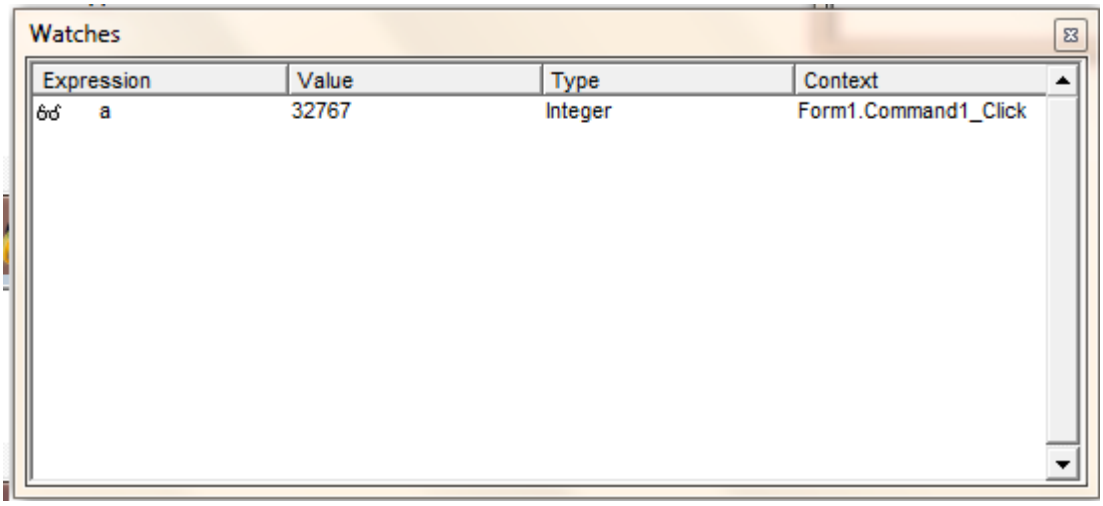

Ta click vào nút *Debug*, chương trình sẽ quay trở lại CS lệnh của VB với một giao diện:




Trong đó, dòng "a=a+1" được hiển thị với màu vàng và 1 mũi tên bên trái, đây chính là dòng và thời điểm mà chương trình gặp lỗi. Ta sử dụng cửa sổ Watch để xem giá trị của biến a. Cách sử dụng như sau: bôi đen biến a, phải chuột tại vị trí đã bôi đen, chọn *Add Watch* từ Shortcut menu, hộp thoại *Add Watch* hiện ra:




Click OK để theo dõi giá trị của biến a trong CS Watches




Cửa sổ Watches cho biết giá trị và kiểu của biến. Trong trường hợp này ta thấy a có giá trị 32767 (đây là giá trị lớn nhất của số Integer), có nghĩa là lệnh a=a+1 sẽ phát sinh lỗi tràn do giá trị vượt quá khả năng lưu trữ của số Integer. Cách sửa lỗi cũng giống ví dụ trước, tức là sử dụng kiểu biến Long có phạm vi giá trị lớn hơn.

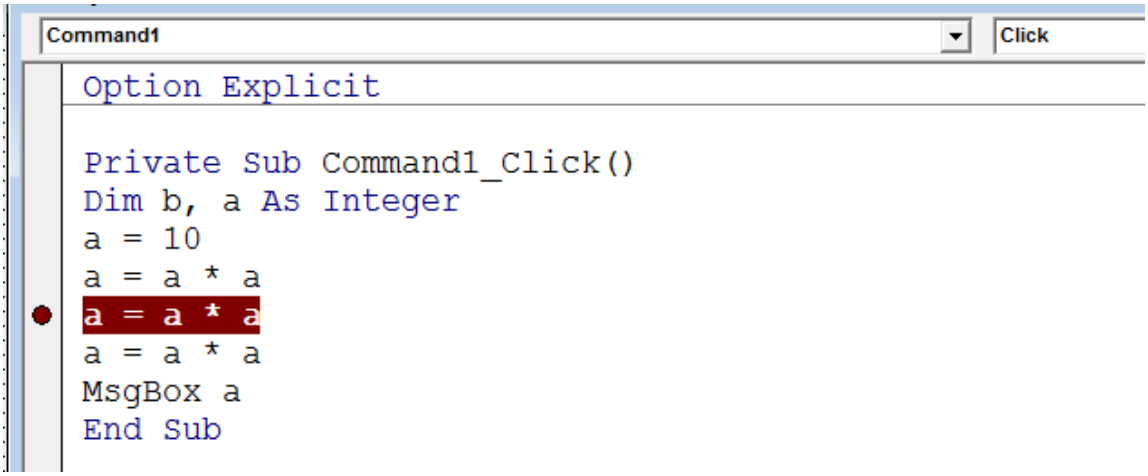
**Chú ý:**

Với các giá trị biến đơn như số, chuỗi... thì có thể xem giá trị của biến bằng cách để con trỏ chuột tại vị trí tên biến, sau khoảng vài giây sẽ có một tooltip hiện lên cho biết giá trị của biến đó. Tuy nhiên cách này không thể sử dụng với biến kiểu mảng, khi đó sử dụng cửa sổ Watch là phương pháp tốt nhất.

### 3) Dùng Breakpoints

Ở phần trên, chúng ta đã dừng chương trình lại để xem giá trị các biến, tuy nhiên việc dừng chương trình là bị động (do có lỗi nên phải dừng lại). Trên thực tế, không phải lúc nào chương trình cũng phát sinh lỗi. Ví dụ như khi kết quả sai do thuật toán lỗi, khi đó phải chủ động dừng chương trình để quan sát giá trị các biến nhằm nắm được nguyên nhân của sai sót.

Ta có thể dùng **Breakpoint** để làm cho chương trình dừng lại ở một chỗ ta muốn ở trong code. Ta cần dự đoán chương trình sẽ đi qua chỗ nào trong code, chọn một chỗ thích hợp rồi click bên trái của hàng code, chỗ dấu chấm tròn đỏ như trong hình dưới đây:



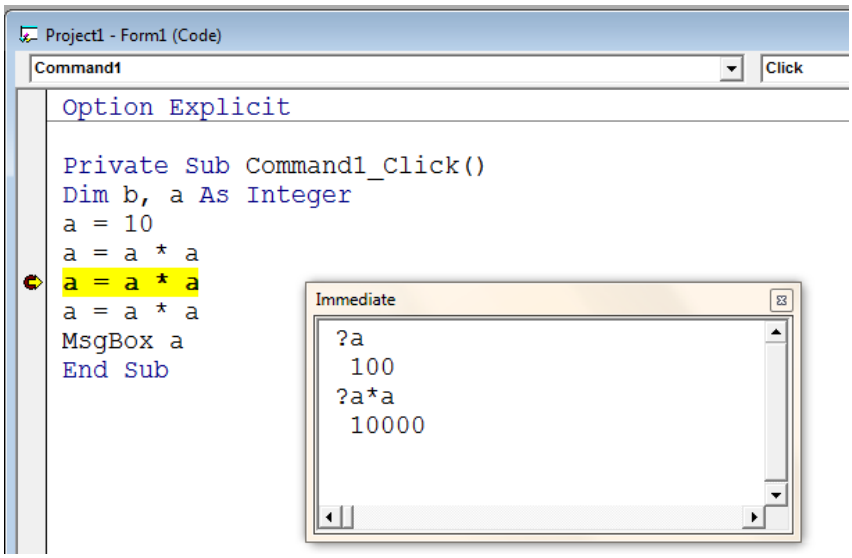
Nếu click lên dấu chấm tròn đỏ một lần nữa thì là hủy bỏ nó. Một cách khác để đặt một breakpoint là đặt con trỏ màn hình lên hàng code rồi bấm **F9**. Nếu bấm F9 lần nữa khi con trỏ màn hình nằm trên hàng đó thì là hủy bỏ break point.

Lúc chương trình đang dừng lại, ta có thể xem giá trị của biến bằng cách để con trỏ màn hình lên trên biến ấy, tooltip sẽ hiện ra giá trị của biến.

Muốn hủy bỏ mọi breakpoints bạn dùng Menu command **Debug | Clear All Breakpoints**.

### 4) Dùng cửa sổ Immediate

Cửa sổ Immediate cho xem giá trị các biến, biểu thức, thực hiện các lệnh VB ngay lập tức trong khi đang gỡ rối chương trình. Khi chương trình đang dừng ở một breakpoint ta có thể thay đổi giá trị của biến hay chạy một lệnh. Để hiển thị Cửa sổ Immediate , dùng Menu command **View | Immediate Window**. Phím bật/tắt cửa sổ Immediate là Ctrl+G. Ví dụ:




Trong cửa sổ Immediate, ta gõ lệnh như minh họa trên, nhấn Enter sẽ nhận được kết quả.

## E./ Nội dung thực hành

### I./ Viết các chương trình con

1) Viết các hàm *USCLN_01, USCLN_02* và *USCLN_03* tính ước số chung lớn nhất của hai số nguyên a, b theo các thuật toán Euclid:
   Hàm *USCLN_01* viết theo thuật toán sử dụng lặp phép trừ.
   Hàm *USCLN_02* viết theo thuật toán sử dụng lặp phép chia.
   Hàm *USCLN_03* viết theo thuật toán đệ quy.
   (Xem hướng dẫn bên dưới)

2) Viết hàm *TongN* tính tổng 1 + 2 + 3 + ... + n.
   (Xem hướng dẫn bên dưới)

3) Nhập 3 số x, y, z. Hoán đổi giá trị giữa chúng để x, y, z tăng dần
   (Xem hướng dẫn bên dưới)

4) Viết thủ tục *TongS* tính tổng 1 + 2 + 3 + ... + n.
(Xem hướng dẫn bên dưới)

5) Viết hàm *KiemTraNT* kiểm tra n có phải là số nguyên tố.
(Xem hướng dẫn bên dưới)

6) Viết hàm *TichN* tính tích 1 * 2 * 3 * ... * n.

7) Viết hàm *TongUS* tính tổng các ước số của n.
Ví dụ: n=10 -> hàm trả về kết quả là 18 (vì có 4 ước số: 1+2+5+10=18)

8) Viết hàm *DemUS* đếm các ước số của n.
Ví dụ: n=10 -> hàm trả về kết quả là 4 (vì có 4 ước số: 1, 2, 5, 10)

9) Viết hàm *KiemTraHH* kiểm tra n có phải là số hoàn hảo.
(Số hoàn hảo là số nguyên dương có tổng các ước số nguyên dương bé hơn nó bằng chính nó.
Ví dụ: 6 là số hoàn hảo vì 6 = 1 + 2 +3. 28 là số hoàn hảo vì 28 = 1 + 2 +4 + 7 +14 ).

10) Viết hàm *KiemTraCP* kiểm tra n có phải là số chính phương.
(Số chính phương hay còn gọi là số hình vuông là số nguyên có căn bậc 2 là một số nguyên, hay nói cách khác, số chính phương là bình phương (lũy thừa bậc 2) của một số nguyên khác. Ví dụ: 4, 9, 16, … là các số chính phương).

11) Viết hàm *USCLN3SO* tính USCLN của 3 số.

12) Viết hàm *TongChuSo* tính tổng các chữ số của số nguyên dương n.
Ví dụ: n=2143 -> hàm trả về kết quả 10 vì 3 + 4 + 1 + 2 = 10

13) Viết hàm *DemChuSo* đếm số lượng chữ số của số nguyên dương n.
Ví dụ: n=2143 -> hàm trả về kết quả: 4

14) Viết hàm *DemNT* đếm các số nguyên tố từ 1->n.
Ví dụ: n=10 -> hàm trả về kết quả là 4 (vì có 4 số nguyên tố: 2, 3, 5, 7).

15) Viết hàm *InNT* in ra n số nguyên tố đầu tiên, mỗi số nguyên tố cách nhau 2 khoảng trắng.
Ví dụ: n=4 -> hàm trả về kết quả là 2 3 5 7
*Function InNT (ByVal n As Integer) As String*

16) Viết hàm đổi số cơ số 10 ra số cơ số 2.
*Function DoiCoSo10Ra2 (n As Integer) As String*

17) Viết hàm đổi số cơ số 2 ra số cơ số 10.
*Function DoiCoSo2Ra10 (n As String) As Integer*

## II./ Một số hướng dẫn bài tập viết chương trình con

1) Viết các hàm *USCLN_01, USCLN_02* và *USCLN_03* tính ước số chung lớn nhất của hai số nguyên a, b theo các thuật toán Euclid:

Trong lý thuyết số, thuật toán Euclid tính ước số chung lớn nhất của hai số nguyên là thuật toán không yêu cầu việc phân tích các số nguyên thành tích các thừa số nguyên tố và nó cũng mang ý

nghĩa lớn vì nó là một trong những thuật toán cổ nhất được biết đến, từ thời Hy Lạp cổ đại. Có hai thuật toán Euclid tính ước số chung lớn nhất của hai số nguyên: Thuật toán sử dụng lặp phép trừ và thuật toán sử dụng lặp phép chia.

a) Hàm *USCLN_01* viết theo thuật toán sử dụng lặp phép trừ.

Xem minh họa trong phần lý thuyết.

b) Hàm *USCLN_02, USCLN_03* viết theo thuật toán sử dụng lặp phép chia:

Cho 2 số tự nhiên a và b, không đồng thời bằng 0: kiểm tra nếu b bằng 0, thì a là ước số chung lớn nhất (USCLN). Nếu không, lặp lại xử lý sử dụng b và a mod b.

$$\text{USCLN}(a,b) = \begin{cases} a & neu\ \ b = 0 \\ \text{USCLN}(b, a \bmod b) & neu\ \ b <> 0 \end{cases}$$

Hướng dẫn viết hàm

```
Private Function USCLN_01(ByVal a As Integer, ByVal b As Integer) As Integer
    Do While a <> b
    If a > b Then
        a = a - b
    Else
        b = b - a
    End If
    Loop
    USCLN_01 = a
End Function
```

```
Private Function USCLN_02(ByVal a As Integer, ByVal b As Integer) As Integer
    Dim temp As Integer
    Do While b <> 0
        temp = b
        b = a Mod b
        a = temp
    Loop
    USCLN_02 = a
End Function
```

```
Private Function USCLN_03(ByVal a As Integer, ByVal b As Integer) As Integer
    If b = 0 Then
        USCLN_03 = a
    Else
        USCLN_03 = USCLN_03(b, a Mod b)
    End If
End Function
```

```
Private Sub CmdTinhUSCLN_Click()
    Dim a As Integer, b As Integer
    a = InputBox("Nhap a")
    b = InputBox("Nhap b")
    MsgBox USCLN_01(a, b)
    a = InputBox("Nhap a")
    b = InputBox("Nhap b")
    MsgBox USCLN_02(a, b)
    a = InputBox("Nhap a")
    b = InputBox("Nhap b")
    MsgBox USCLN_03(a, b) '
End Sub
```

*Chú ý:*

Nếu nhập a bằng 2 tỷ và b bằng 2 thì bạn hãy cho biết vòng lặp trong các hàm *USCLN_01* và *USCLN_02* lặp bao nhiêu lần.

2) Viết hàm *TongN* tính tổng 1 + 2 + 3 + ... + n.

Hướng dẫn: cho biến i chạy từ 1 đến n đồng thời sử dụng cộng dồn bằng công thức s=s+i

Chương trình:

```
Private Function TongN(ByVal n As Integer) As Long
    Dim i As Integer, s As Long
    For i = 1 To n
        s = s + i
    Next
    TongN = s
End Function
```

```
Private Sub CmdTongN_Click()
    Dim n As Integer
    n = InputBox("Nhap n")
    MsgBox TongN(n)
End Sub
```

Chú ý: có thể dùng vòng lặp Do While Loop (hoặc Do Loop Until), cho n giảm dần về 1 đồng thời sử dụng cộng dồn bằng công thức s = s + n

```
Private Function TongN_C2(ByVal n As Integer) As Long
    Dim s As Long
    Do While n >= 1
        s = s + n
        n = n - 1
    Loop
    TongN_C2 = s
End Function
```

3) Nhập 3 số x, y, z. Hoán đổi giá trị giữa chúng để x, y, z tăng dần

Hướng dẫn:

So sánh x với y, nếu x>y thì hoán đổi giá trị của x và y.

So sánh x với z, nếu x>z thì hoán đổi giá trị của x và z.

So sánh y với z, nếu y>z thì hoán đổi giá trị của y và z.

Chương trình:

```
Private Sub Swap(ByRef a As Integer, ByRef b As Integer)
    Dim Temp As Integer
    Temp = a
    a = b
    b = Temp
End Sub
```

```
Private Sub CmdSapxep3so_Click()
    Dim x As Integer, y As Integer, z As Integer
    x = InputBox("Nhap x")
    y = InputBox("Nhap y")
    z = InputBox("Nhap z")
    If x > y Then Call Swap(x, y)
    If x > z Then Call Swap(x, z)
    If y > z Then Swap y, z  'Swap y, z tuong duong Call Swap(y, z)
    MsgBox x & Space(5) & y & Space(5) & z
End Sub
```

Chú ý:

Trong thủ tục *Swap*, bạn hãy thay cách truyền tham số bằng từ khóa *Byval* và chạy lại chương trình, quan sát kết quả để rút ra nhận xét cần thiết.

4) Viết thủ tục *TongS* tính tổng 1 + 2 + 3 + ... + n.

Hướng dẫn:

Khi cần dùng thủ tục để tính và lưu kết quả thì ta phải dùng tham số hình thức dạng tham biến (*ByRef*) để lưu lại kết quả cần tính.

Chương trình:

```
Private Sub TongS(ByRef s As Long, ByVal n As Integer)
    Dim i As Integer
    For i = 1 To n
        s = s + i
    Next
End Sub
```

```
Private Sub CmdTongS_Click()
    Dim n As Integer, s As Long
    n = InputBox("Nhap n")
    Call TongS(s, n)
    MsgBox s
End Sub
```

5) Viết hàm *KiemTraNT* kiểm tra n có phải là số nguyên tố

Số nguyên tố là số tự nhiên chỉ chia hết cho 1 và chính nó. Ngoài ra nó không chia hết cho bất cứ số nào khác. Số 0 và 1 không được coi là số nguyên tố.

Gợi ý:

+ Cách 1: Đếm tất cả các ước của n, nếu biến đếm = 2 thì n là số nguyên tố. Để đếm các ước của n ta cho i chạy từ 1 đến n và kiểm tra i có là ước của n.

+ Cách 2: Cho i = 2, lặp tăng i cho đến khi i là ước của n. Nếu i = n thì n là số nguyên tố.

+ Một hàm có chức năng kiểm tra đúng sai thì kiểu dữ liệu của hàm phải là kiểu Boolean trả về trị *True*, *False*.

Chương trình:

```
Private Function KiemTraNT_C1(ByVal n As Integer) As Boolean
    Dim i As Integer, Dem As Integer
    If n <= 1 Then
        KiemTraNT_C1 = False
        Exit Function
    End If
    For i = 1 To n
        If n Mod i = 0 Then Dem = Dem + 1
    Next
    If Dem = 2 Then
        KiemTraNT_C1 = True
    Else
        KiemTraNT_C1 = False
    End If
End Function
```

```
Private Function KiemTraNT_C2(ByVal n As Integer) As Boolean
    Dim i As Integer
    If n <= 1 Then
        KiemTraNT_C2 = False
        Exit Function
    End If
    i = 2
    Do While n Mod i <> 0
        i = i + 1
    Loop
    If n = i Then
        KiemTraNT_C2 = True
    Else
        KiemTraNT_C2 = False
    End If
End Function
```

```
Private Sub CmdKiemTraNT_Click()
    Dim n As Integer
    n = InputBox("Nhap n")
    If KiemTraNT_C1(n) Then
        MsgBox n & " la so nguyen to"
    Else
        MsgBox n & " khong la so nguyen to"
    End If

    'KiemTraNT_C2
    n = InputBox("Nhap n")
    If KiemTraNT_C2(n) Then
        MsgBox n & " la so nguyen to"
    Else
        MsgBox n & " khong la so nguyen to"
    End If
End Sub
```

Chú ý: Nếu nhập n bằng 2 tỷ thì bạn hãy cho biết vòng lặp trong các hàm *KiemTraNT_C1* và *KiemTraNT_C2* lặp bao nhiêu lần.

6) Viết hàm *TichN* tính tích 1 * 2 * 3 * ... * n.

Hướng dẫn: cho biến i chạy từ 1 đến n đồng thời sử dụng nhân dồn bằng công thức s = s * i. Chú ý biến s phải khởi động bằng 1.

7) Viết hàm *TongUS* tính tổng các ước số của n.

Hướng dẫn: cho biến i chạy từ 1 đến n, với mỗi i là ước của n thì sử dụng cộng dồn bằng công thức s = s + i.

8) Viết hàm *DemUS* đếm các ước số của n.

Hướng dẫn: cho biến i chạy từ 1 đến n, với mỗi i là ước của n thì tăng biến đếm.

9) Viết hàm *KiemTraHH* kiểm tra n có phải là số hoàn hảo.

Hướng dẫn: cho biến i chạy từ 1 đến n\2, với mỗi i là ước của n thì sử dụng cộng dồn bằng công thức s = s + i. Nếu n = s thì n là số hoàn hảo.

10) Viết hàm *KiemTraCP* kiểm tra n có phải là số chính phương.

Hướng dẫn:

+ Hàm Sqr(n) cho căn bậc 2 của n.

+ Hàm Round(Sqr(n)) làm tròn đến phần nguyên của căn bậc 2 của n

+ Nếu Sqr(n) = Round(Sqr(n)) thì n là số chính phương.

11) Viết hàm *USCLN3SO* tính USCLN của 3 số.

Hướng dẫn:

+ Cách 1:

Nhập 3 số a, b, c

d=USCLN(a, b)

USCLN3SO=USCLN(d, c)

+ Cách 2:

Nhập 3 số a, b, c.

d=Min(a, b, c)

Cho biến i chạy từ 1 đến d, với mỗi i là ước đồng thời của a, b, c thì USCLN3SO = i.

12) Viết hàm *TongChuSo* tính tổng các chữ số của số nguyên dương n.

Hướng dẫn: Qua ví dụ n=2143 -> hàm trả về kết quả 10 vì 3 + 4 + 1 + 2 = 10 ta lần lượt lấy ra các con số từ phải qua trái. Cách lấy:

Do While (n= 0)

Sodu = n Mod 10

n = n \ 10

Cộng dồn bằng công thức s = s + Sodu

Loop

13) Viết hàm *DemChuSo* đếm số lượng chữ số của số nguyên dương n.

Hướng dẫn: Ta lần lượt lấy ra các con số từ phải qua trái rồi tăng biến đếm.

## III./ Chạy tay chương trình VB sử dụng chương trình con

### 1) Ví dụ 1

**a) Cho chương trình:**

```
Private Sub p(ByVal x As Integer, ByRef y As Integer)
        x = 3 * y - x
        y = x + y
        MsgBox "x = " & x & Space(5) & " y = " & y
End Sub

Private Sub CmdVidu1_Click()
        Dim a As Integer
        Dim b As Integer
        a = 5
        b = 2
        MsgBox "a =" & a & Space(5) & "b =" & b
        p a, b
        MsgBox "a = " & a & Space(5) & " b = " & b
End Sub
```

**b) Cách trình bày chạy từng bước chương trình trên giấy:**

| Câu lệnh | Các biến và giá trị của chúng | | | | Màn hình |
|---|---|---|---|---|---|
| Dim a As Integer<br>Dim b As Integer | a | b | | | |
| a = 5<br>b = 2 | 5 | 2 | | | |
| MsgBox "a=" & a & Space(5) & "b =" & b | | | | | a=5 b=2 |
| p a, b<br>p(ByVal x As Integer, ByRef y As Integer) | | | x<br>5 | y | |
| x = 3 * y - x<br>y = x + y | | 3 | 1 | | |
| MsgBox "x =" & x & Space(5) & "y =" & y | | | | | x=1 y=3 |
| Kết thúc chương trình con | Biến x và y bị xóa khỏi bộ nhớ | | | | |
| MsgBox "a =" & a & Space(5) & "b =" & b | | | | | a=5 b=3 |

## 2) Ví dụ 2

**a) Cho chương trình:**

```
Private Sub S(ByRef a As Integer, ByVal b As Integer)
        a = a + b
        b = 2 * b - a
End Sub

Private Sub CmdVidu2_Click()
        Dim a As Integer
        Dim b As Integer
        a = 4
        b = 5
        MsgBox "a=" & a & Space(5) & "b=" & b
        Call S(b, a)
        MsgBox "a=" & a & Space(5) & "b=" & b
End Sub
```

**b) Cách trình bày chạy từng bước chương trình trên giấy:**

| Câu lệnh | Các biến và giá trị của chúng | | | | Màn hình |
|---|---|---|---|---|---|
| Dim a As Integer<br>Dim b As Integer | a | b | | | |
| a = 4<br>b = 5 | 4 | 5 | | | |
| MsgBox "a=" & a & Space(5) & "b=" & b | | | | | a=4 b=5 |
| Call S(b, a)<br>S(ByRef a As Integer, ByVal b As Integer) | | | a | b<br>4 | |
| a = a + b<br>b = 2 * b - a | | 9 | | -1 | |
| Kết thúc chương trình con | Biến tham số a và b bị xóa khỏi bộ nhớ | | | | |
| MsgBox "a =" & a & Space(5) & "b =" & b | | | | | a=4 b=9 |

### 3) Ví dụ 3

**a) Cho chương trình:**

```
Private Sub S(ByRef a As Integer, ByVal b As Integer)
        a = a + b
        b = 2 * b - a
End Sub

Private Sub CmdVidu3_Click()
        Dim i As Integer
        Dim a As Integer
        Dim b As Integer
        i = 6
        a = 4
        b = 5
        MsgBox "a=" & a & Space(5) & "b=" & b
        Do While i >= 5
            If i <= 5 Then
               S b, b
            Else
               Call S(a, a)
            End If
            i = i - 1
        Loop
```

MsgBox "a=" & a & Space(5) & "b=" & b
End Sub

**b) Cách trình bày chạy từng bước chương trình trên giấy:**

| Câu lệnh | Các biến và giá trị của chúng | | | | | Màn hình |
|---|---|---|---|---|---|---|
| Dim i As Integer<br>Dim a As Integer<br>Dim b As Integer | i | a | b | | | |
| i = 6<br>a = 4<br>b = 5 | 6 | 4 | 5 | | | |
| MsgBox "a=" & a & Space(5) & "b=" & b | | | | | | a=4 b=5 |
| Do While i >= 5 → (ĐK đúng, lặp lần 1)<br>If i <= 5 ... → (ĐK sai, gọi lệnh sau Else) | | | | | | |
| Call S(a, a)<br>S(ByRef a As Integer, ByVal b As Integer) | | | | a<br>4 | b | |
| a = a + b<br>b = 2 * b - a | | 8 | | | 0 | |
| Kết thúc chương trình con S(a, a) | Biến tham số a và b bị xóa khỏi bộ nhớ | | | | | |
| i = i - 1 | 5 | | | | | |
| Do While i >= 5 → (ĐK đúng, lặp lần 2)<br>If i <= 5 ... → (ĐK đúng, gọi lệnh sau Then) | | | | | | |
| S b, b<br>S(ByRef a As Integer, ByVal b As Integer) | | | | a | b<br>5 | |
| a = a + b<br>b = 2 * b - a | | | 10 | | 0 | |
| Kết thúc chương trình con S b, b | Biến tham số a và b bị xóa khỏi bộ nhớ | | | | | |
| i = i - 1 | 4 | | | | | |
| Do While i >= 5 → (ĐK sai, kết thúc Lặp) | | | | | | |
| MsgBox "a =" & a & Space(5) & "b =" & b | | | | | | a=8 b=10 |