6: Subprogramas

Subprogramas

1. Introducción a los Subprogramas
2. Tipos y uso de subprogramas
3. Parámetros
4. Tipo de parámetros: Por Valor/Por Referencia
5. Especificación

Ejercicios

Introducción (I)

A la hora de resolver un problema:

- Dividir Problema en subproblemas más pequeños
- Resolver cada subproblema por su lado
- Los subproblemas a su vez se pueden dividir en otros subproblemas
- Así hasta que los subproblemas a resolver sean fáciles de resolver

Introducción (II)

Ejemplo: Dados los números positivos a, b donde a >=b

$$\binom{a}{b} = \frac{a!}{b!(a-b)!}$$

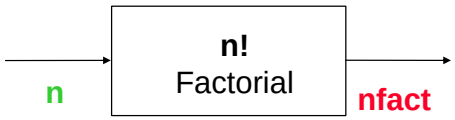
- Algoritmo:
  1. lee los valores a y b
  2. calcula el resultado
  3. escribe su resultado
- Algoritmo:
  2. calcula el resultado
  2.1 a_fact = a!
  2.2 b_fact = b!
  2.3 a_menos_b_fact = (a-b)!
  2.4 resultado = a_fact / (b_fact * a_menos_b_fact)

Introducción (III)

- Los problemas 2.1, 2.2, 2.3 son iguales:

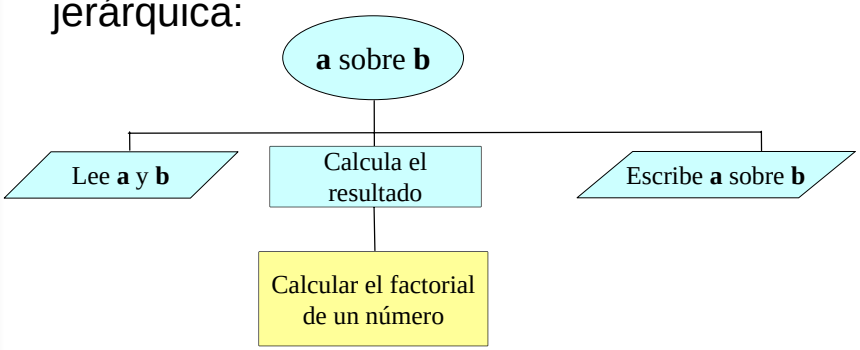
  Calcular **n!**, factorial de n, y dejar el resultado en una variable nfact

```
nfact =1
For i=1 To n Step 1
   nfact = nfact * i
Next i
```



Introducción (IV)

- La solución organizada de manera jerárquica:



Motivación

- Modulación:
  - Cada subprograma tiene una misión bien definida
- Ahorro de memoria y tiempo de desarrollo:
  - Se reduce el número de líneas de código
  - La probabilidad de cometer errores se reduce considerablemente
- Independencia de datos y ocultamiento de la información:
  - Es independiente de los otros subprogramas
  - Mantiene sus propios datos
  - Define una interfaz sencilla para la comunicación con el programa.
  - No tiene acceso a la información que no necesita
- Facilidad de mantener
  - Cuando repetimos el mismo código numerosas veces si hay que modificarlo supone muchas modificaciones, aumentando la posibilidad de error.

Tipos de Subprogramas

- En VB existen 2 tipos:
  - **Función**:
    - La llamada a una función **siempre devuelve un valor**
    - Se puede utilizar el valor devuelto dentro de una expresión o asignar a una variable directamente
    - En ejecución, la llamada a la función se sustituye por el valor que devuelve dicha función
    - Ejemplo: `y = sin(x)`
  - **Procedimiento**:
    - La llamada a un procedimiento **no devuelve ningún valor**. Por lo tanto no se puede utilizar dentro de una expresión.
    - Se pueden cambiar el valor de los parámetros ,en caso de que se quiera devolver algún valor.(No es obligatorio)

¿Cuándo utilizar procedimientos o cuándo funciones?

Operaciones	Funciones	Procedimiento
Lee de teclado	Sí	Sí
Escribir en pantalla	No	Sí
No devuelve ningún resultado	No	Sí
Devuelve un resultado	Sí	Sí
Devuelve varios resultados	No	Sí

Definición de procedimientos

- Sintaxis:

  [`Private`] `Sub` *Nombre* [`(` *Parámetros* `)`]
  [*Declaración de variables locales*]
  [*Instrucciones*]
  `End Sub`

- Ejemplo:

```
Sub saludo (ByVal nombre As String)
  MsgBox ("Hola " & nombre)
End Sub
```


Definición de funciones

- Sintaxis:

  [`Private`] `Function` ***Nombre*** (*[Parámetros]*) [`As` *Tipo*]
  [*Declaración de variables locales*]
  [*Instrucciones*]
  Nombre = *Expresión*
  `End Function`

- Ejemplo:

```
Function cuadrado (ByVal x As Integer) As Integer
  cuadrado = x * x
End Funcion
```


Definición de funciones: ejemplo

```
Function factorial(ByVal n As Integer) As Long
   Dim i As Integer
   Dim nfact As Long

   nfact = 1
   For i=1 To n Step 1
      nfact = nfact * i
   Next i
   factorial = nfact
End Function
```

- Llamada:

```
Dim a_fact As Long
a_fact = factorial(a)
```

Definición de procedimientos: ejemplo

```
Private Sub factorial(ByVal n As Integer, nfact As Long)
   Dim i As Integer

   nfact = 1
   For i=1 To n Step 1
      nfact = nfact * i
   Next i
End Sub
```

- Llamada:

```
Dim a_fact As Long
Call factorial(a, a_fact)
```


Ejemplo: definición de procedimientos y llamada a funciones (utilización)

```
Private Sub cmdCalcular_Click()
   Dim a As Integer, B As Integer
   Dim resultado As Single
   Dim a_fact As Long
   Dim b_fact As Long
   Dim a_menos_a_fact As Long

   a = InputBox("Introduce un número:")
   b = InputBox("Introduce un número:")
   a_fact  = Factorial(A)
   b_fact  = Factorial(B)
   a_menos_b_fact = Factorial(a-b)
   resultado = a_fact / (a_Fact * a_menos_b_fact)
   MsgBox("El resultado de la combinación : " & resultado)
End Sub
```


Ejemplo: procedimiento con parámetros

```
Private Sub polinomio (ByVal a As Integer, ByVal b As Integer, _
                       ByVal c As Integer, _
                       x1 As Single, x2 As Single, _
                       grado As Integer, reales As Boolean)
  Dim discriminante As Long

  reales = True
  If a = 0 Then
    If b = 0 Then
      grado = 0
    Else
      grado = 1
      x1 = -c / b
    End If
  Else
    discriminante = b ^ 2 - 4 * a * c
    grado = 2
    If discriminante >= 0 Then
      x1 = (-b + Sqr(discriminante)) / 2 * a
      x2 = (-b - Sqr(discriminante)) / 2 * a
    Else
      reales = False
    End If
  End If
End Sub
```


Llamada a Procedimientos: [Call]

```
Private Sub cmdRaices_Click()
  Dim c1 As Integer, c2 As Integer, c3 As Integer
  Dim rdo1 As Integer, rdo2 As Integer, reales As Boolean

   c1 = InputBox("Introduce un número:")
   c2 = InputBox("Introduce un número :")
   c3 = InputBox("Introduce un número :")
   Call polinomio(c1,c2,c3,rdo1,rdo2,grado,reales) ' Call opcional
   If reales Then
     If grado=2 Then
       MsgBox("raíces:" & x1 & " y " & x2)
     ElseIf grado=1 Then
       MsgBox("La raíz es:" & x1)
     Else
       MsgBox("No tiene raíces")
     End If
   End If
End Sub
```

La ejecución en una llamada a procedimiento

Los de la definición de la función

a. Antes de la llamada
- Los parámetros formales no tienen valor

b. Llamada a procedimiento
- Se reserva espacio para almacenar los parámetros y las variables locales del procedimiento
- Los parámetros reales se evaluan, y a cada parametro formal se le asigna su parametro real

Los de la llamada a la función

c. Ejecución del subprograma

d. Retorno al programa principal

e. Desaparecen todas las variables reservadas por el subprograma

Parámetros: paso de valores

- Los parámetros se utilizan para intercambiar valores entre el programa principal y el subprograma
- En VB existen 2 tipos de parámetros:
  - Entrada: únicamente cuando se quiere pasar valores del programa principal al subprograma. En la definición se utiliza `ByVal` (**paso por valor**).
  - Entrada-Salida: Además de poderse utilizar como entrada, pueden devolver un resultado al programa principal. Se trabaja directamente con la variable original de llamada. No funciona si pasamos expresiones.

Parámetros: paso de valores (II)

- A la hora de pasar valores como parametros por valor al subprograma, estos pueden ser:
  - Valores constantes: `5`, `3.4`
  - El contenido de una variable: `num`
  - Una expresión: `num+5`
- Cuando queremos que las modificaciones sobre el parámetro afecten a la variable de la llamada (parámetro de salida):
  - Variable

Parametros: paso de valores (III)

- Por convenio, vamos a definir en la cabecera del subprograma:
  - los parametros de entrada al principio
  - los de entrada/salida al final
- Ejemplo:

```
Private Sub polinomio(ByVal a As Integer, _
    ByVal b As Integer, ByVal c As Integer, _
    x1 As Single, x2 As Single, grado As Integer, _
    reales As Boolean)
```

- Gráficamente:

Parametro. Paso de Valores(IV)

- Dada la siguiente definición de procedimiento:

```
Sub Cal_Max_Min (ByVal Num1 As Integer, _
                 ByVal Num2 As Integer, _
                 Max As Integer, _
                 Min As Integer)
```

- Para utilizarlo dentro del programa principal, sabiendo que las variables del programa principal son: A, B, Maximo, Minimo. La llamada quedaría:

```
Call Cal_Max_Min(A,B,Maximo,Minimo)
```

- Dada la siguiente definición de función:

```
Function Factorial (ByVal X As Integer) As Long
```

- Para utilizarlo dentro del programa principal, sabiendo que las variables del programa principal son: Y y Z. La llamada quedaría:

```
Y = Factorial(Z)
```


Parametro:¿Por referencia o por copia?

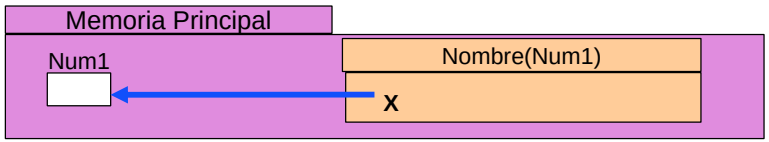
- Por referencia:
  - Lo que se copia es la referencia a memoria de la variable ,es decir, la variable utilizada en la llamada y su variable correspondiente en el subprograma, apuntan a la misma celda en memoria.

Ejemplo:

Definición: `Sub nombre(X As Integer)`

Llamada: `Call nombre(Num1)`

No es obligatorio



Parametros:¿Por referencia o por copia? (I)

- Por referencia:
  - Si dentro del subprograma se cambia el valor del parametro formal, también se cambiará su correspondiente parametro real, es decir, su valor en el programa principal.
  - En VB por defecto se pasa por referencia
  - La **tablas** (arrays) y las matrices **siempre** se pasan **por referencia**

Parametros:¿Por referencia o por copia?(II)

- Por Copia:
  - Se **copia el valor** del parámetro real (programa principal) al parámetro formal (definido en el subprograma)
  - 2 opciones:
    1. En la llamada añadiendo parentesis:

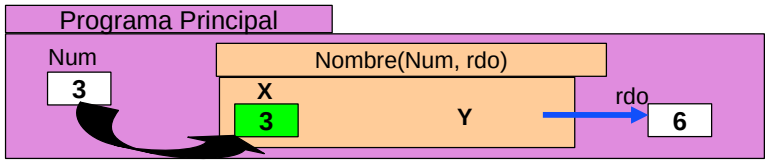
```
Call nombre ((Num), rdo)
```

    1. En la definición con **ByVal** :

```
Sub nombre(ByVal X As Integer, _
                Y As Long)
```

Parametros:¿Por referencia o por copia?(III)

- Por Copia:
  - Se usan cuando no queremos modificar el valor introducido como parámetro. (Ejemplo: aunque se cambie el valor de X en el subprograma la variable Num no cambia)



Especificación(I)

Formalmente:

- CABECERA + CONDICIONES

CABECERA

- Existen tres elementos principales:
  1. **Tipo** de subprograma: *Function*, *Sub*
  2. **Nombre** del subprograma: *Absoluto*
  3. **Declaración de los parámetros** (parecido a la declaración de variables):
     a. Al principio se ponen las parámetros de entrada y luego los de entrada-salida
     b. Por cada parámetro:
        - **Tipo (E o E/S): `ByVal`** o nada
        - **Nombre**: *Num*
        - **Tipo_variable**: `Integer`
- Ejemplo:

```
Function Absoluto (ByVal Num As Integer) As Integer
```


Especificación(II)

CONDICIÓN

- Hay que añadir 2 condiciones (con comentarios):
- **Precondición** (la condición previa a la ejecución del subprograma):
  - **Condiciones** que deben **cumplir los parámetros de entrada**
  - Si dentro del procedimiento se **lee algún dato por teclado,** se comenta aqui.
- **Postcondición** (condición que se da una vez finalizada la ejecución del subprograma):
  - Se comentan los **posibles valores de los parámetros de salida**
  - Si el procedimiento **muestra algún mensaje o valor por pantalla** se comenta aqui.

Especificación(III)

- Ejemplos:

```
Function Calcular_Max(ByVal Z1 As Integer, _
                      ByVal Z2 As Integer) As Integer
  ' Precondición: Z1 y Z2 son valores enteros
  ' Postcondición: devuelve el mayor entre Z1 y Z2

Sub Calcular_Max_Min(ByVal Num1 As Integer, _
                     ByVal Num2 As Integer, _
                     ByVal Num3 As Integer, _
                     Max As Integer, Min As Integer)
  ' Precondición: pasamos 3 enteros
  ' Postcondición: Max devuelve el mayor de Num1, Num2, Num3
  '                Min devuelve el menor de Num1, Num2, Num3
```

Especificación(III)

- Ejemplos:

```
Sub Calcular_Suma(ByVal tope As Integer, _
                  ByVal lim As Integer, _
                  T() As Integer, _
                  suma As Integer)
  ' Precondición:
  ' tope: número de elementos introducidos
  ' lim: índice máximo de la tabla suponiendo que
  '      el índice de inicio es cero
  ' T: tabla
  ' Postcondición: devuelve en suma el
  ' sumatorio de los valores de la tabla
```


Ejercicios de Especificación

- Escribe la especificación de los siguientes subprogramas:
  1. Realizar un subprograma que nos diga si un número es **primo** o no.
  2. Realizar un subprograma que nos calcule el sumatorio de un número dado N (1+2+3+...+N)
  3. Realizar un subprograma que lea por teclado una secuencia de números terminados por cero y nos muestre por pantalla cuantos y cuales son los números primos introducidos.