# Programación modular con LabVIEW*

Patxi Alkorta Egiguren

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 2.0†

**Abstract**

Este módulo explica y proporciona la información necesaria para poder generar módulos o funciones en LabVIEW.

## 1 Programación modular

Como en el resto de los lenguajes de programación, también en **LabVIEW**, el concepto de programación modular descansa sobre el elemento función. Esto quiere decir que la programación modular se basa en la programación con funciones, es decir, que la función es la base de la programación modular. Muchas de estas funciones son proporcionadas por el propio fabricante de software, y se presentan ordenadas por tipos en las librerías de funciones. Estas librería pueden ser vistas si se pulsa con el botón derecho del ratón sobre cualquier punto del diagrama de nuestra aplicación, a continuación se pulsa en la opción final del menú flotante (doble ^ o extensión), con lo que se muestra el listado de las librerías de funciones. La mayoría de las funciones que utilizaremos en este curso, serán las de la librería **Programming** (programación), Figure 1.

*Version 1.1: Oct 24, 2008 11:42 am -0500

†http://creativecommons.org/licenses/by/2.0/

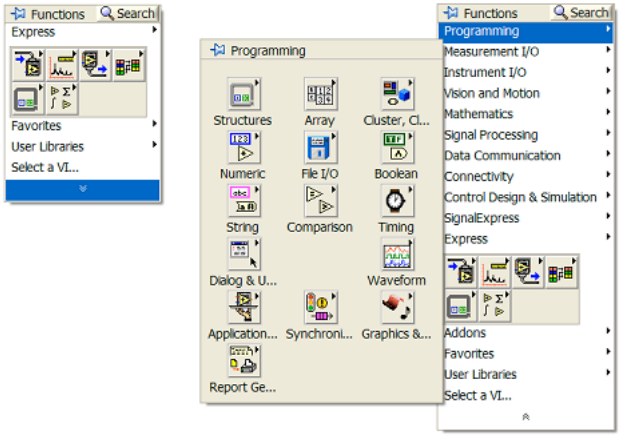


**Figure 1**

Como se puede observar, si se despliega esta librería, las funciones aparecen ordenadas por tipos. Por ejemplo, en el grupo **Structures** (estructuras), nos encontramos con las sentencias de repetición **while** y **for**, las sentencias de secuenciación, las de selección, etc, con las que trabajaremos más adelante.

Por otro lado, el programador tiene la capacidad de generar por sí solo las funciones que necesita para el desarrollo de sus aplicaciones. Cuando una misma tarea es requerida en varios puntos de la aplicación, en lugar de construir esa porción del código (diagrama) de forma repetida, lo que se suele hacer es construirla una vez y llamarla en todos los puntos de la aplicación que sea necesaria. Además, una vez construidas o realizadas las funciones, el programador las puede utilizar en otros proyectos de aplicaciones futuras. Incluso las puede mejorar y/o cambiar para utilizarlas en otras aplicaciones.

## 2 Creando un SubVI

Una función en el lenguaje de programación **LabVIEW**, es conocida con el nombre de **SubVI** (sub instrumento virtual). Veamos a continuación, cómo se construye una **SubVI** y lo fácil que resulta utilizarla. Para ello, se utilizará el ejercicio propuesto **Calculadora.vi** del módulo anterior y se guardará como **Calculadora1.vi**. En primer lugar se crea el código con el que se quiere hacer el **SubVI**, a continuación se marca esa porción o bloque de código con el ratón y se ejecuta **CreateSubVI**, desde el menú **Edit**, con lo que el código marcado queda asociado al nuevo **SubVI** con el icono standard de **LabVIEW**, Figure 2. Para cambiar este icono, se hace doble click sobre él, donde se abre el archivo del **SubVI** y se ejecuta **Ctrl+T** para ver tanto el panel como el diagrama de esta función. Tanto en el panel como en el diagrama, las salidas aparecen nombradas por defecto con **Numeric** (izda), cuando lo que interesa es nombrarlas según su función (dcha), ya que de lo contrario cuando vayamos a utilizar esta función no sabremos qué proporciona en cada una de las salidas, Figure 3.

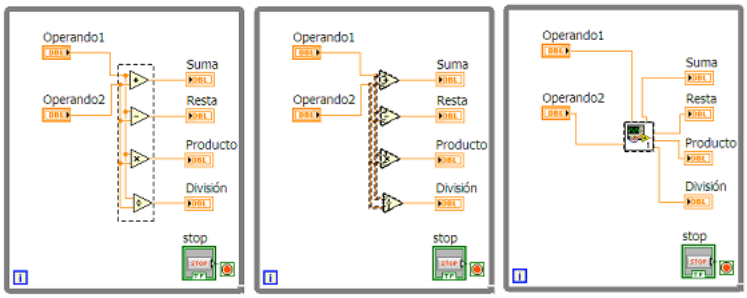


**Figure 2**

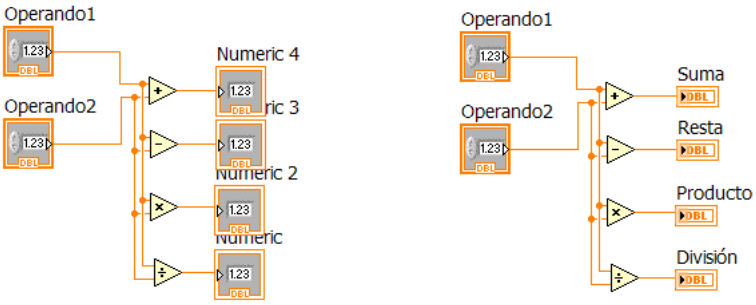


**Figure 3**

A continuación, se pulsa el botón derecho de ratón sobre el símbolo del icono (arriba a la derecha del diagrama) y se ejecuta **Edit Icon**, tras lo cual aparece la ventana de definición del icono del **SubVI**, Figure 4. Este editor de iconos es muy fácil de utilizar, ya que es muy intuitivo. Para eliminar el contenido del icono por defecto, se marca todo el contenido del mismo con la herramienta selección, habiéndo seleccionado previamente en la paleta de herramientas, y pulsando **Supr**. A partir de ahí, se puede escribir desde el teclado, para lo cual ha de seleccionarse la herramienta texto , o también se puede dibujar, utilizando la herramienta lápiz, Figure 5.

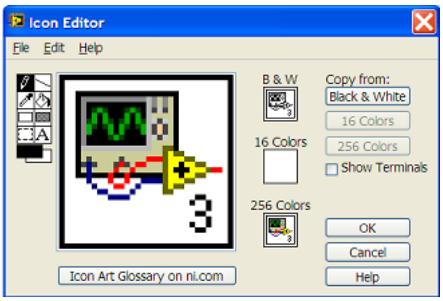


**Figure 4:** Editor de iconos de LabVIEW.

**Figure 5:** Herramientas de selección, lápiz y texto.

Para el caso que nos sigue, se utiliza la herramienta lápiz, con el que se dibujan los símbolos de las cuatro operaciones aritméticas, Figure 6. Tras ello se pulsa **OK**, donde se puede observar que el icono que aparece ahora tanto en esta función como en la aplicación donde la hemos creado (**Calculadora1.vi**) es el definido por nosotros. A continuación, se guarda el archivo **SubVI** con el nombre **Calc4 (SubVI).vi**. Notar que el sufijo **SubVI** que aparece entre paréntesis, en el nombre de la función, es insertado por la herramienta **LabVIEW** de forma automática.

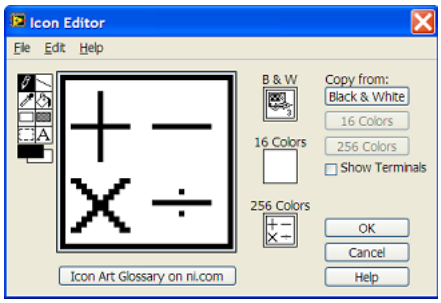


**Figure 6**

El aspecto final que mostraría la aplicación utilizando el **SubVI Calc4** sería el que aparece en la Figure 7 izquierda. Conviene comentar que esta función ha sido creada desde la aplicación **Calculadora1.vi**, pero a partir de ahora, cuando es insertada para ser utilizada en alguna otra aplicación, habrá que identificar cada

una de las entradas y salidas, por lo que tendrá que ser expandida desactivando su opción **View As Icon**, Figure 7 derecha. Ésto también es válido para las funciones de librería de **LabVIEW**.

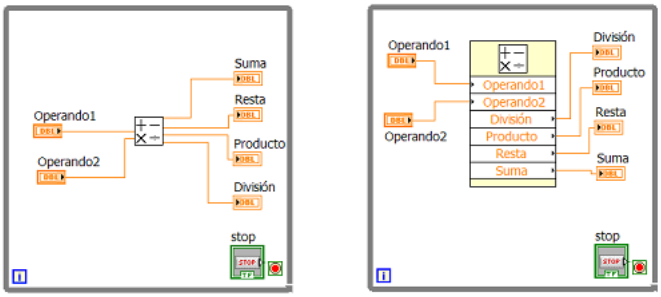


**Figure 7**

A partir de ahora, cada vez que se desee insertar cualquier **SubVI** previamente definido en cualquier aplicación que tengamos abierta en ese momento, basta con ejecutar **Select a VI**, del menú flotante de **Functions** del diagrama, Figure 8. En realidad, esta forma sirve para introducir tanto **SubVI**s como **VI**s, en la aplicación abierta. Es conveniente comentar, que en este caso concreto, en un principio podríamos pensar que se podía haber introducido el **VI Calculadora1.vi** directamente, el ejercicio propuesto en el módulo anterior. Lo que ocurre es que tal y como se planteó, éste se ejecuta dentro de una sentencia **while**, por lo que a nosostros nos sobra esta sentencia y por ello no nos sirve incluir este proyecto directamete. En el caso de que ese proyecto no incluyera la sentencia **while**, entonces sí hubiese servido de **SubVI**.

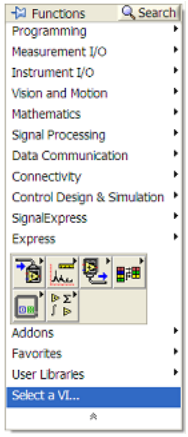


**Figure 8**

Hay que añadir, que la generación de iconos personalizados no está restringida solamente a **SubVI**s, también es posible hacerlo en cuanquier **VI**.

De esta manera, utilzando las **SubVI**s, es posible establecer varios niveles de llamadas entre **SubVI**s, al

igual que ocurre con las funciones en los lenguajes de programación estructurada: una función es llamada desde una aplicación, y ésta a su vez llama a otra y así sucesivamente. Ejecutando **View/VI Hierarchy** es posible visualizar la jerarquía de llamadas entre **SubVI**s de una palicación dada. La Figure 9 muestra la jerarquía del ejemplo que nos ocupa. Notar que el nivel superior es el entorno de desarrollo **LabVIEW** de la aplicación **Calculadora1.vi**, y el inferior **Calc4.vi**.

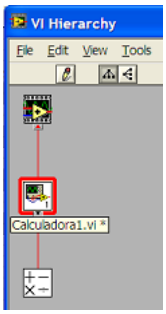


**Figure 9:** Jerarquía de la nueva aplicación Calculadora1.vi

## 3 La ayuda de LabVIEW

La herramienta **LabVIEW** dispone de una potente ayuda que facilita enormemente la tarea de programar. Cuando tenemos alguna duda, nos la puede aclarar acudiendo a ella. Cuando queremos ampliar nuestros conocimientos sobre las posibilidades que tiene la herramienta, también podemos acudir a ella, ya que en realidad se trata de un libro en soporte informático. A continuación se van a describir las opciones más utilizadas de la ayuda de esta herramienta: el menú **Help**, accesible tanto desde el panel como del diagrama.

**Help/Show Help Context**, al ejecutarla, esta opción queda marcada y ello indica que la ventana de contexto (**Context Help**) está abierta. Esta ventana muestra la información resumida del objeto sobre el cual se sitúa el cursor del ratón, funciona tanto con objetos del panel como con los del diagrama. La Figure 10 muestra la información sobre el objeto divisor de dos operandos.

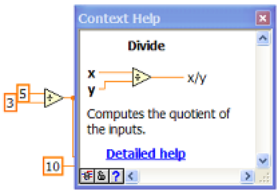


**Figure 10**

Si se quiere información más detallada, se pulsa sobre el link **Detailed help**, dentro de la ventana de contexto, y a continuación aparece otra ventana, la de ayuda de **LabVIEW** (**LabVIEW Help**), donde se muestra la información detallada, en este caso, de la función división, Figure 11.

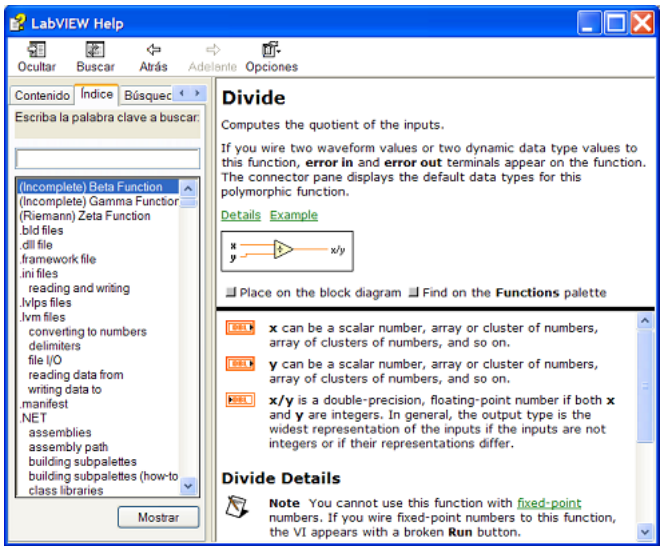


**Figure 11**

**Help/Search the LabVIEW Help**..., al ejecutar esta opción, aparece la ventana anterior **LabVIEW Help**. En su parte izquierda muestra tres lengüetas. Así, **Contenido**, como su nombre indica, muestra la información de las capacidades de **LabVIEW** organizada por temas y subtemas. **Índice**, sirve para realizar búsquedas por todo el contenido de la ayuda, y **Búsqueda**, para hacer una búsqueda más profunda.

Justo sobre la mitad de la ventana, en el centro, aparece el botón **Place on the block diagram**, así, si se pulsa sobre él, el objeto o función del que se muestra la información es colocado sobre el diagrama, sin tener que saber en qué submenú de **Functions** se encuentra exactamente. Ahora bien, si se desea saber dónde se encuentra exactamente (para ir conociendo la paleta de las funciones), se pulsará sobre el botón **Find on the Functions palette**, donde a continuación aparecerá la ventana **Functions**. Siguiendo con el ejemplo de la función **Divide**, si se desea saber dónde está exactamente esta función, al pulsar el mencionado botón, aparece la ventana de Figure 12. Así podemos desplazarnos hacia arriba a través de esta paleta, mediante el coamdo flecha hacia arriba, y conocer exactamente su ubicación en ella.

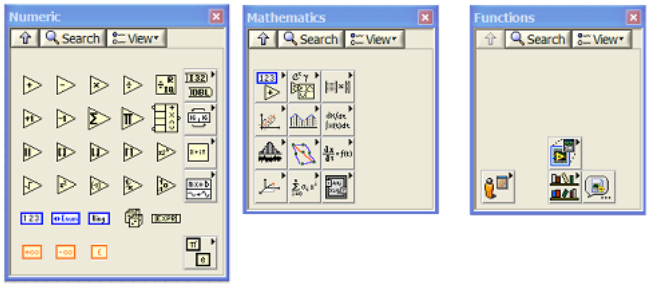


**Figure 12**

## 4 Ejercicios propuestos

**Exercise 1**

Explora en la paleta de funciones **Functions** los menús **Programming** y **Mathematics** e intenta comprender las funciones, una a una, que aparecen en ellas. Utiliza para ello el conocimiento que tienes de haber cursado las asignaturas de tu carrera. Seguro que hay funciones que las identificarás nada más verlas, y otras, aunque te lleven algún tiempo, también las acabarás identificando, aunque posiblemente no las identifiques todas. Intenta también, escribiendo en una hoja, dibujar el arbol organizativo de los menús que se te piden conocer.

**Exercise 2**

Realizar un **VI**, **Calculadora2.vi**, que contenga a un **SubVI** llamado **AriTriRaInv.vi** y que realice 3 tipos de operaciones definidas en 3 **SubVI**s de un nivel inferior: **SubVI Calc4.vi** que realiza las cuatro operaciones aritméticas con dos operandos (1 y 2) y que proporciona un resultado por operación, **SubVI Trig4.vi** que realiza las cuatro operaciones trigonométricas de seno, coseno, tangente y cotangente con el operando 1 con el ángulo en grados, y que también proporciona un resultado por operación, y, **SubVI RaiztInv.vi**, que realiza las operaciones de raíz cuadrada e inversa con el operando 2, y que proporciona un resultado por operación. Desplegar en pantalla la jerarquía de **SubVI**s y meditar sobre el resultado.

**Exercise 3**

Busca y asimila la información, utilizando la ayuda de **LabVIEW**, acerca de las operaciones con matrices (matrix) suma (add), resta (substract), producto (product), determinante (determinant), inversa (inverse) y transpuesta (transpose). Esta información será de gran ayuda para realizar el último ejercicio del tema de los arrays y clusters.