

---

# **PROGRAMACIÓN ESTRUTURADA EN DFD**

---

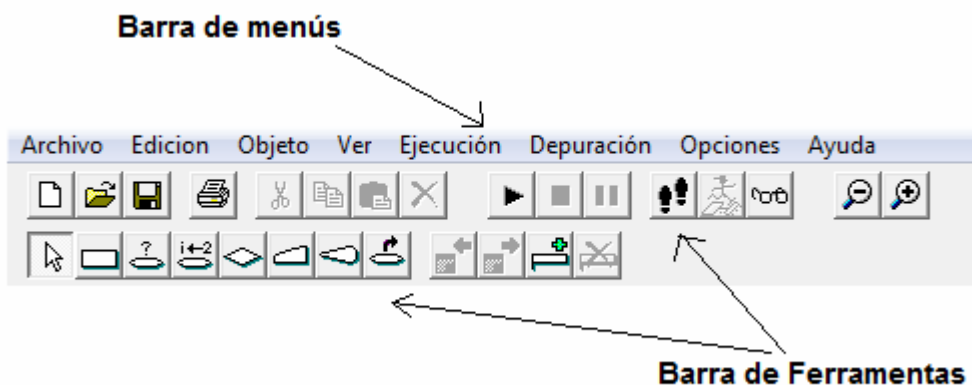
**IX XORNADAS DE TECNOLOXÍA DE GALICIA**

# 1. INTRODUCCIÓN

Un diagrama de fluxo de datos e unha descripción gráfica dun procedemento para a resolución dun problema.

Os diagramas de fluxo de datos están formados por figuras conectadas con frechas. Para executar un proceso descrito por un diagrama de fluxo de datos empézase polo INICIO e continúaase polas frechas de figura en figura, executándose as accións indicadas por cada figura; o tipo de figura indica unha determinada sentença .

Os compoñentes básicos da ventá principal son: A **barra de menú**, **barras de ferramentas**, **barras de desprazamento laterais** e a **área de traballo**.



Dfd é un software deseñado para construír e analizar algoritmos. Pódense crear diagramas de fluxo para a representación de algoritmos de programación estruturada a partires das ferramentas de edición que para este propósito trae o programa. Despois de ingresar o algoritmo representado polo diagrama, pódese executar, analizar e depurar nun entorno interactivo deseñado para este fin.

A hora de sentarse a programar debemos ter en conta que:

- DFD non distingue entre maiúsculas e minúsculas.
- DFD non almacena datos nin xera arquivos executables.
- DFD non compila. (Xerar arquivos en BINARIO ou HEX).
- Non entende de faltas de ortografía.

O que si fai DFD:

- Detecta posibles erros no código. (Fase de depuración)
- Executa o programa paso a paso (step to step) para ver a evolución do mesmo.

Por outra parte, debemos distinguir entre o “**programador**” que é a persoa que crea o programa, o “**usuario**” que é a persoa que vai utilizar dito programa e por último, o **ordenador** onde vai ser executado ese software.

## 2. VARIABLES E CONSTANTES EN DFD.

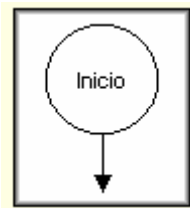
En DFD pódense distinguir dous tipos de datos: As “**variables**” e as “**constantes**”. Unha variable é un símbolo que pode tomar infinitos valores. Por exemplo, a variable “SUMA” nun intre dun programa pode valer 10 e máis tarde 40. Pola contra unha constante, sendo tamén un símbolo, unicamente ten un valor independentemente do momento que sexa invocada polo programa. Un exemplo é “ $\pi$ ” que sempre vai valer 3.1416....

O nome das variables en DFD son escollidas polo programador, e como norma xeral, deberán ter unha relación co programa. Un exemplo, si calculamos unha superficie dun CADRADO a variable que recolle este valor podería chamarse “ÁREA ou “SUPERF”.

## SÍMBOLOS PARA PROGRAMAR EN DFD

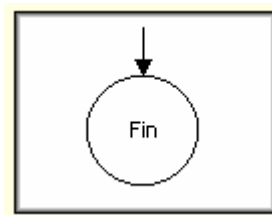
### Inicio

É o primeiro símbolo a executar en calquera algoritmo. Ó ser executado, o Inicio transfere o control ó seguinte símbolo.



### Fin

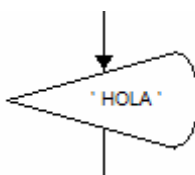
Este símbolo, xunto co símbolo Inicio, delimita o corpo do programa principal. Sólo existe un símbolo Fin no diagrama; a execución deste objeto finaliza a execución do programa.



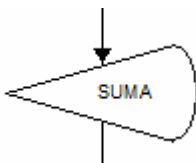
### Saída de datos

O símbolo **Saída** mostra valores por pantalla. En certa maneira, a forma deste símbolo recorda a un monitor CRT visto de perfil. Permite visualizar “variables” ou “comentarios”.

Cando necesitamos representar un comentario este debe ir con comillas simples. No exemplo emitimos unha mensaxe o usuario de benvida.



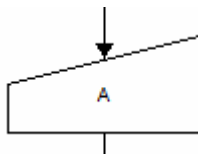
Cando unha variable se representa SIN comillas simples visualizamos O VALOR NUMÉRICO de dita variable. Neste caso o valor da variable SUMA.



Para poder teclear a variable ou o comentario debemos facer dobre clic co punteiro do rato no símbolo unha vez colocado no lugar axeitado.

### Lectura de datos.

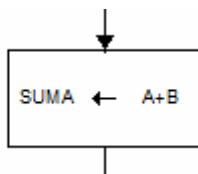
O símbolo **Lectura** permite a entrada de valores desde o teclado por parte do usuario (en certa maneira este símbolo podería asemellarse a un teclado un pouco máis longo). O executarse, o símbolo desplega un cadro de diálogo por cada variable presente na lista, este cadro de diálogo espera que o usuario introduzca un valor que será asignado a respectiva variable. No exemplo, recollida dun valor para a variable A.



Para ingresar as variables débese facer dobre clic no símbolo.

### Asignación de variables.

O símbolo Asignación asigna valores a variables. Normalmente realiza operacións matemáticas. Ó ser executado, pode realizar hasta tres asignacións. No exemplo, asigna á variable SUMA os valores da variable A sumados á variable B.



O cadro de diálogo do símbolo Asignación contén espacio para tres asignacións, cada asignación consta dun espacio para o campo variable situado sempre a esquerda, o símbolo de asignación e un espacio para a expresión situada sempre á dereita. Isto indica que o campo variable se lle asigna o resultado da avaliación da expresión.

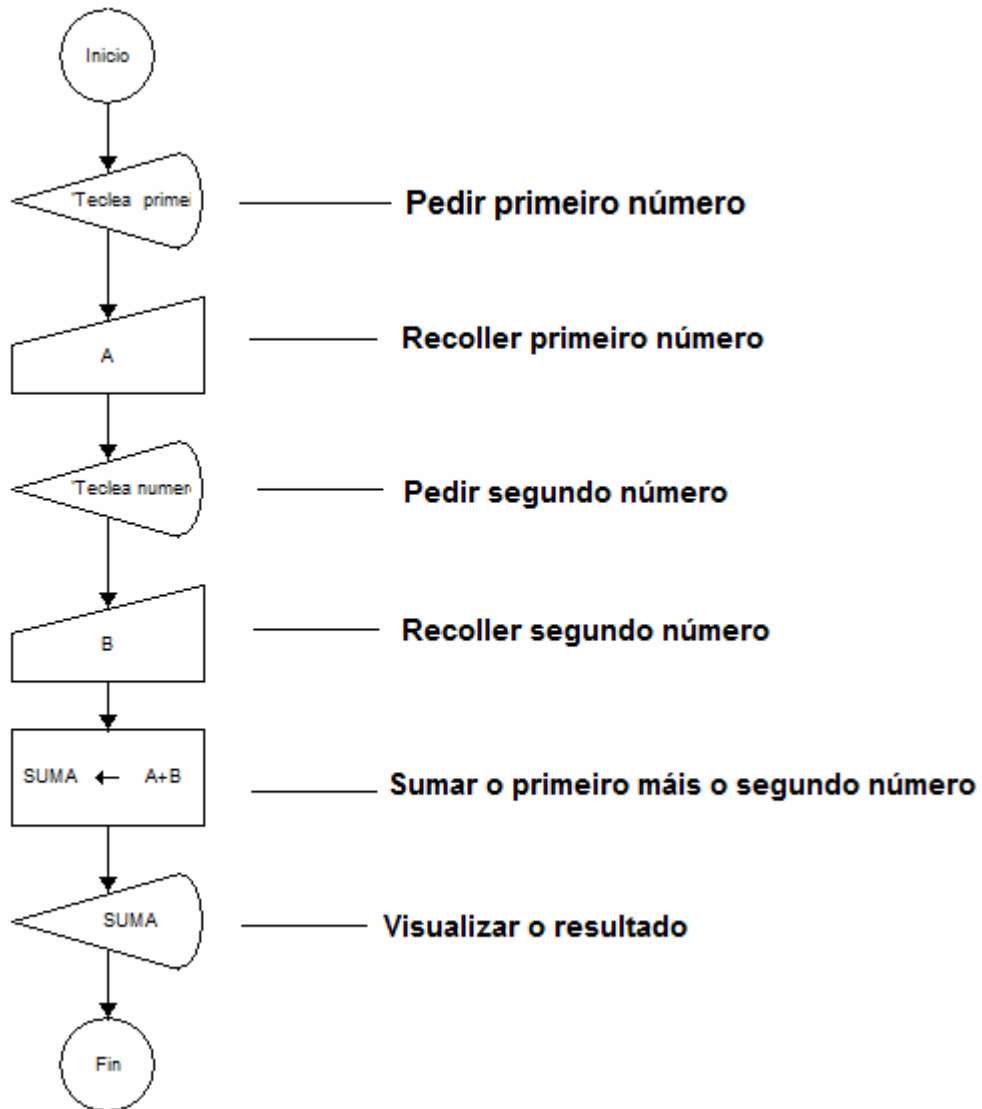
## EXERCICIOS PROPOSTOS (Nivel I)

Como a programar apréndese programando ahí van uns cuantos :

1. Diseñar un programa que haga a suma de dous números introducidos por teclado.

2. Diseñar un programa que calcule a área e máis o perímetro dun cadrado. O usuario tecleará o lado.
3. Programa que calcula os anos de idade dunha persoa, para iso teclearase polo usuario o ano de nacemento.
4. Programa que faga a conversión de Euros a pesetas. O usuario tecleará unha cantidade en Euros.

A modo de exemplo, resólvese o primeiro deles:



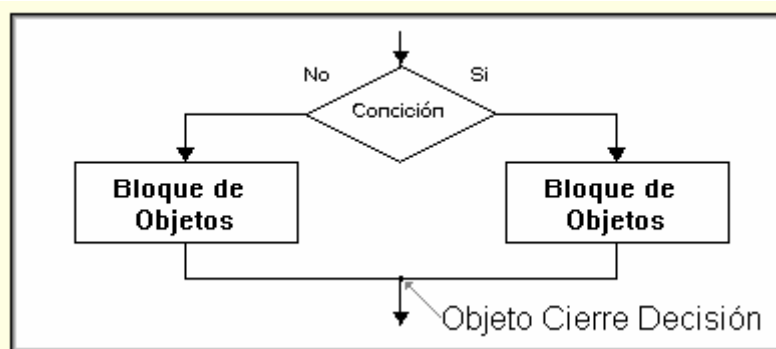
Para comprobar o seu funcionamento débese premer na barra de ferramentas o botón “Executar ”



### 3. BUCLES

Moitas veces os programas evalúan variables internas que fan redirixir o fluxo de datos en función da avaliación dunha ou varias variables. Encontrámonos cos bucles.

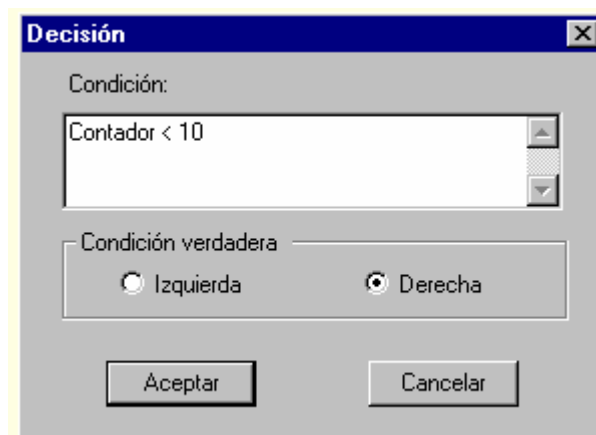
#### Decisión (Bucle “if”).



O símbolo decisión selecciona o fluxo a seguir dacordo co valor lóxico dunha condición. A condición debe ser sempre unha expresión que ó ser avaliada dé como resultado un valor de tipo de dato Lóxico.

Exemplo :  $3 < w$  ,  $x > 0$  ,  $\text{valor} * 15 < 300 * \text{contador}$ .

Si o avaliar a condición obtense o valor lóxico .V., execútase o bloque rotulado ca palabra “Si”; en caso contrario, execútase o bloque rotulado con “No”.



O cadro de diálogo Decisión contén espazo para a expresión que conforma a condición, e dúas casillas por medio das cales pódese especificar por que lado continuará o fluxo en caso de que a condición sexa verdadeira.

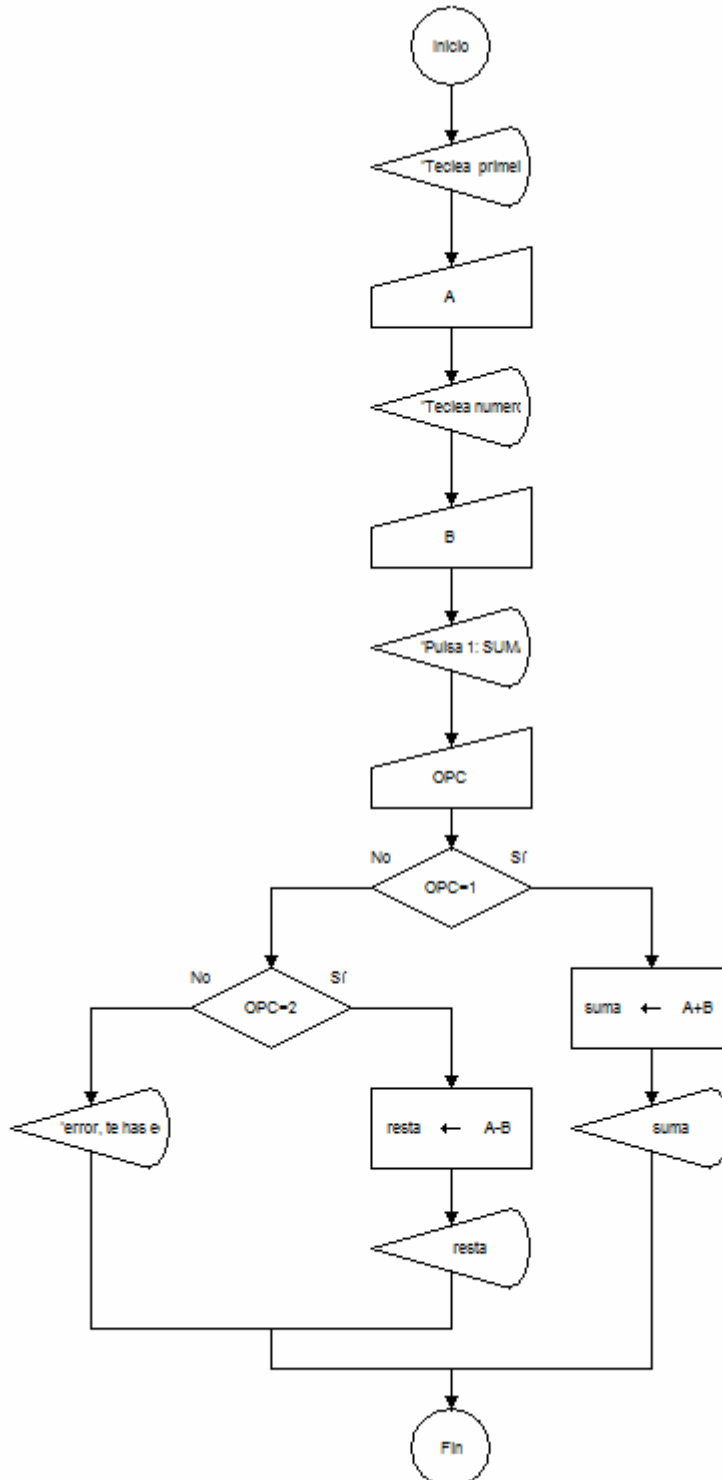
#### EXERCICIOS PROPOSTOS (Nivel I – Bucles “if”)

1. Programa que determina si un número introducido por teclado é positivo ou negativo.
2. Programa que comproba si o segundo número introducido por teclado é CERO. En caso afirmativo emitírase unha mensaxe de error. En caso negativo realízase a división deses dous números.
3. Programa que determina o maior de dous números introducidos por teclado.

Pode ser que o programa conteña varios bucles anidados formando unha escaleira condicional.

Vexamos cun exemplo:

1. Diseñar un programa que faga a SUMA ou a RESTA de dous números introducidos por teclado. O usuario poderá escoller a operación a realizar.

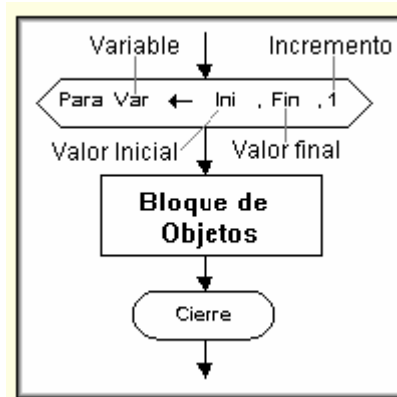


**EXERCICIOS PROPOSTOS (Nivel I – Bucles “if”)**

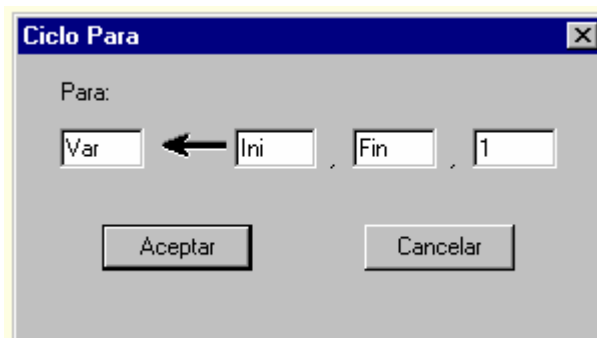
1. Diseñar unha calculadora básica (sumar, restar, multiplicar e dividir)
2. Diseñar unha calculadora científica (sumar, restar, multiplicar e dividir e funcións trigonométricas).
3. Diseñar un programa que nos indique si un número é positivo, negativo ou nulo.

## Bucle Para (Bucle “for”).

A súa función é executar un bloque de símbolos mentres que a variable contadora non supere o límite establecido polo valor final.



Contén ademais un valor inicial que será asignado ó contador o iniciar a execución do ciclo, un valor final e un valor de incremento. Si o contador excede o valor final, a execución continuará a partires do símbolo que sigue o Cierre. En caso contrario, executarase o corpo do ciclo e o contador será incrementado no valor indicado polo incremento.

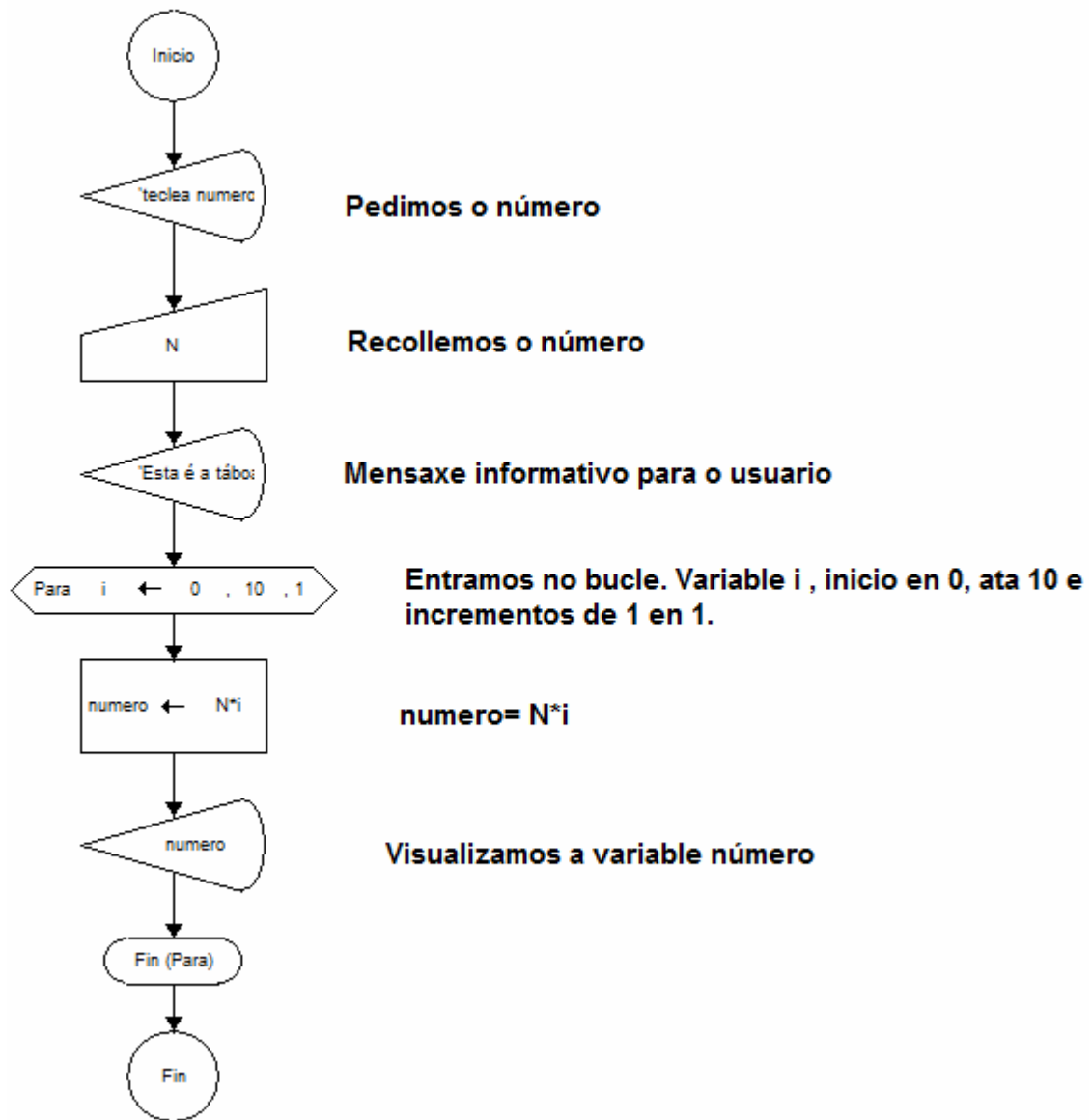


O cadro de diálogo do **Ciclo para** contén espacio para a variable contador, valor inicial, valor final e o valor de incremento no seu respectivo orde.

### EXERCICIOS PROPOSTOS (Nivel I – Bucles “for”)

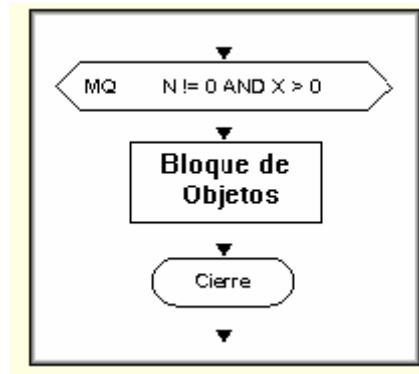
1. Deseñar un programa que faga a suma dunha determinada cantidade de números introducidos por teclado. O usuario tecleará a cantidade de números.
2. Deseñar unha calculadora básica que teña incluído o programa anterior.
3. Deseñar un programa que represente a táboa de multiplicar dun número. O usuario tecleará o número do cal quere obter a táboa.

Resolvemos o número 3 a modo de exemplo.



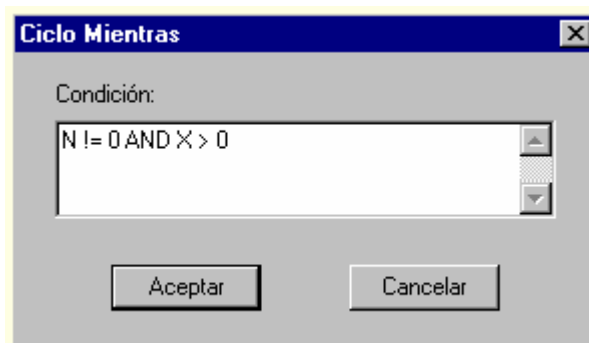
Agora, a xeito de prácticas, deixamos en mans do programador o exercicio 1 e 2.

## Bucle Mientras (Bucle “do-while”)



O símbolo “**Ciclo Mientras**” ten como función executar un bloque de símbolos mentres que unha condición sexa verdadeira. A condición avalíase sempre o principio do bucle e para que se volva a executar o bucle a variable deberá ser de novo avaliada e ser V (verdadeira).

Si o avaliar a condición obtense o valor .F (falso) a execución do algoritmo continuará a partires do símbolo que sigue o Cierre.



O cadro de dialogo do símbolo Ciclo Mientras contén espacio para a expresión que conforma a condición de permanencia.

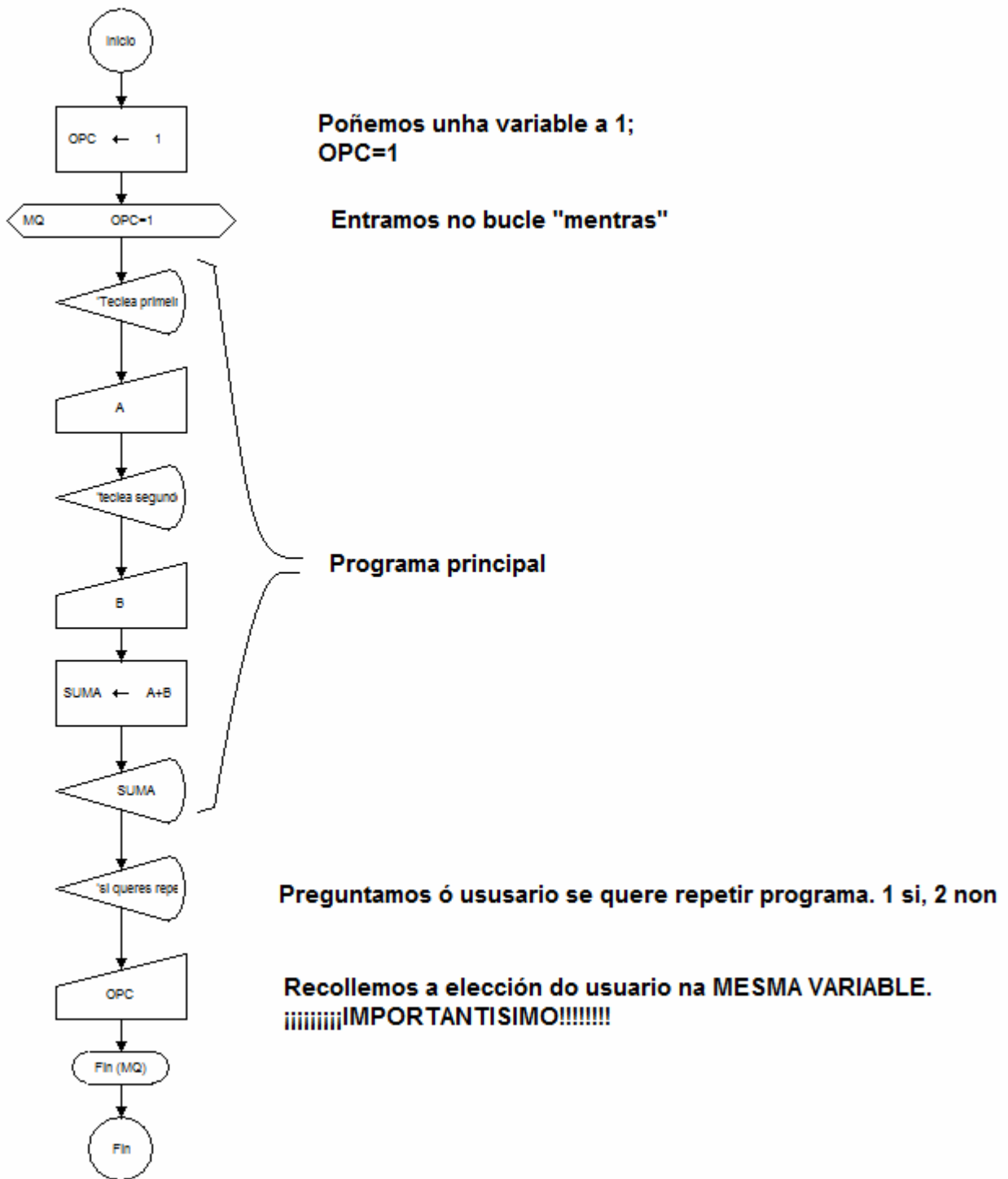
O bucle Mientras ten moitísimas utilidades : substituir un bucle para, crear programas con contraseñas de usuario, bucles repetitivos, etc..

A modo de exemplo, ilústrase un programa repetitivo (repítese o programa o número de veces que o desexe o usuario).

### EXERCICIOS PROPOSTOS (Nivel I – Bucles “do-While”)

1. Diseñar un programa repetitivo que faga a suma de dous números introducidos por teclado
2. Diseñar unha calculadora básica que sexa repetitiva.
3. Diseñar un programa repetitivo que calcule a área dunha circunferencia.

Resolvemos o primeiro a modo de exemplo:

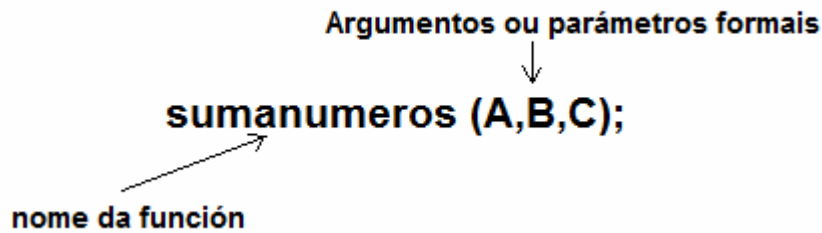


## 4. FUNCIONES OU SUBPROGRAMAS

Unha función ou subprograma é un trozo de código que NON pertence o programa principal pero que resolve unha pequena parte do mesmo. Trátase de trocear un programa grande (que ocuparía moita pantalla e moitos recursos do PC ou memoria) en programas máis pequenos (subprogramas). Pódense crear infinitas funcións, incluso o programa principal pode ser un subprograma doutro programa. O número final de funcións depende de cada programador e do complicado que sexa o programa principal.

Cando unha función chama a outra función fálase de **recursividade**.

Unha función ten as seguintes partes:



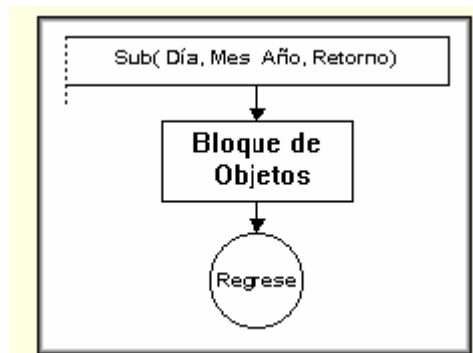
Existen dous tipos de funcións:

- Funcións de librería: Están feitas polas persoas que crearon DFD; por exemplo TAN(X), SIN(X), COS(X), ARCOS(X), SQRT(X),..... (remítovos a axuda do programa )
- Funcións deseñadas polo usuario: Resolven unha necesidade do programa en cuestión e son propias de cada caso. Exemplos: Recolle\_datos(A,B,C) , Sumatodo(A,B) , etc...



Subprograma

### Símbolo Subprograma



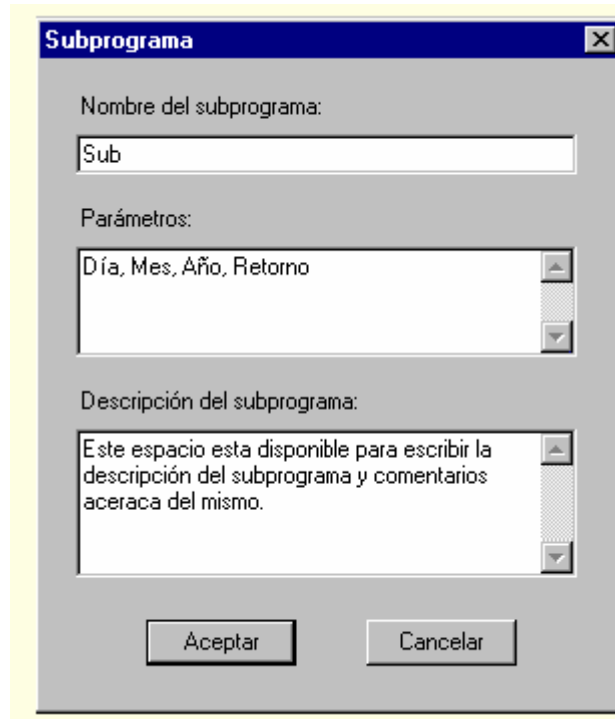
É o primeiro símbolo en ser executado cando un subprograma é chamado. O ser executado, o símbolo Subprograma transfere o control ó símbolo interior do subprograma.

O cadro de diálogo do símbolo Subprograma contén un espazo para a descrición ou comentarios acerca do mesmo; contén un espazo para o nome do subprograma e un espazo para os parámetros

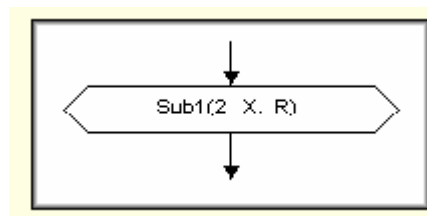
formais. Estes parámetros formais (si existen) deben estar separados por comas. O nome dun subprograma debe comezar por unha letra seguida de letras, números ou o carácter ( \_ ).

Exemplo: Suma\_datos, calcula\_area, ....

Non se ten en conta a diferenza entre maiúsculas e minúsculas para o nome dun subprograma, é dicir, SUB equivale a sub.



**Símbolo “Chamada a subprograma”.**

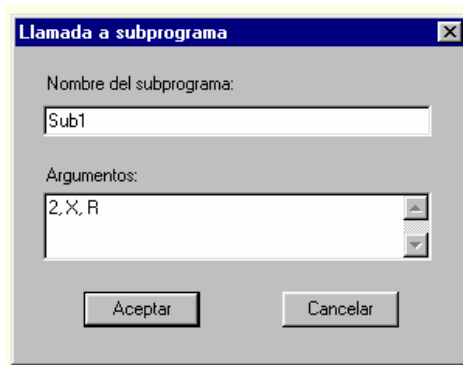


A función deste símbolo é realizar unha chamada a un subprograma. Na chamada deben encontrarse os argumentos que han de ser pasados ó subprograma, a cantidade, a orde e o tipo dos argumentos deben coincidir cos parámetros do subprograma.

Unha vez que o subprograma sexa executado a execución continuará no símbolo seguinte á chamada.



**Chamada a Subprograma**



O cadro de diálogo para a edición deste símbolo contén o espazo para o nome do subprograma a chamar e o espazo para a lista de argumentos. Ditos argumentos deben estar separados por comas.

### EXERCICIOS PROPOSTOS (Nivel II-Subprogramas)

Resolver, cada programa, utilizando funcións:

1. Diseñar un programa que faga a **CONVERSIÓN DE EUROS A PTS.**
2. Diseñar un programa que calcule o **PERÍMETRO** dun **RECTÁNGULO** e máis a **ÁREA**. O usuario deberá teclear os lados.
3. Diseñar un programa que calcule o **PERÍMETRO** dun **CADRADO** e máis a **ÁREA**. O usuario deberá teclear os lados.
4. Diseñar un programa que halle o **PERÍMETRO** dun **TRIÁNGULO** e máis a súa **ÁREA**. O usuario deberá teclear os lados.
5. Diseñar un programa que calcule a **MEDIA de 15** números introducidos por teclado.
6. Diseñar un programa que faga a **CONVERSIÓN** de **Km/h** a **m/s**.
7. Diseñar un programa que resolva unha **ECUACIÓN** de **2º grao**. Supoñer que ten dúas solucións.
8. Diseñar un programa que pida un número a través de pantalla. O usuario deberá teclear dito número e o programa entregará o produto dese número por 4 e a división dese número entre 2.
9. Diseñar un programa que, mediante a Lei de Ohm, calcule a tensión nunha resistencia coñecendo a Intensidade.
10. Diseñar un programa que calcule a temperatura en graos Fahrenheit e máis en Graos Kelvin, coñecendo a temperatura en graos Celsius.

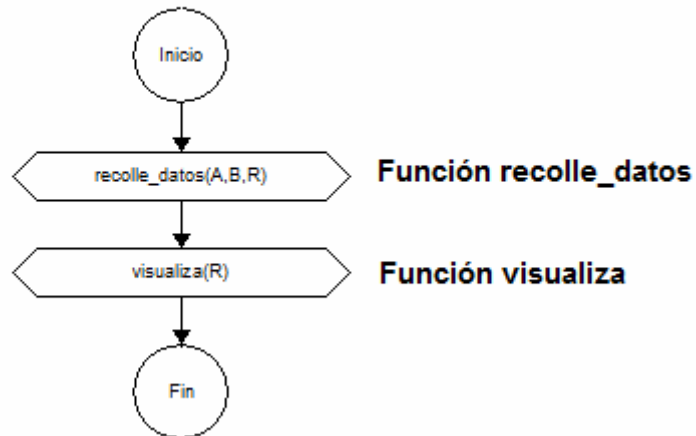
$$^{\circ}\text{F} = (9/5^{\circ}\text{C} + 32)$$

$$\text{K} = ^{\circ}\text{C} + 273$$

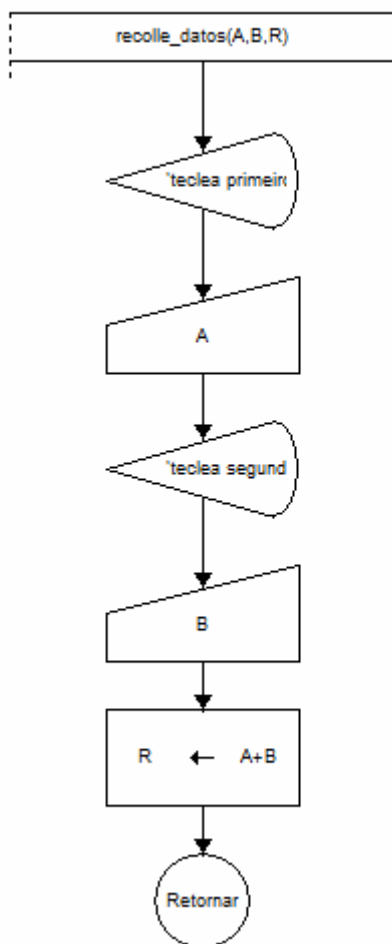
11. Diseñar un programa que determine o **VOLUME** dunha pirámide de base **CUADRANGULAR**.
12. Diseñar un programa que determine o **VOLUME** dun **CILINDRO**.
13. Crear un programa que pida o radio dunha **CIRCUNFERENCIA** e que calcule a lonxitude da mesma, así como a superficie interior.
14. Diseñar un programa que determine o **VOLUME** dunha **SEMIESFERA**.

Exemplo: Utilizando funcións, deseñar un programa que determine a suma de dous números introducidos por teclado.

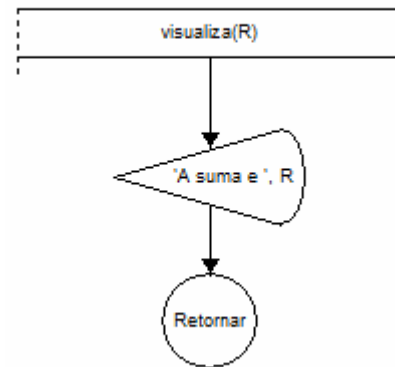
**Programa principal**



**Función recolle\_datos**

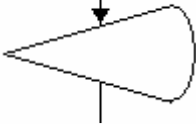
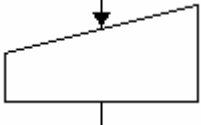
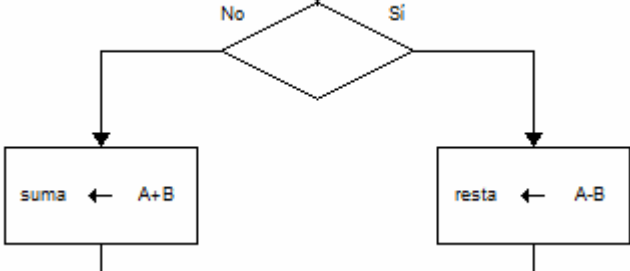
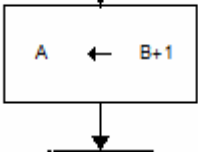
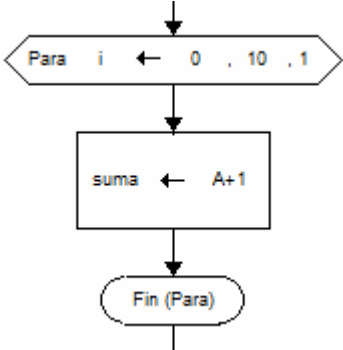


**Función visualiza**



ANEXO I- Programación estructurada con Arduino.

Do mesmo xeito que un programa se pode executar en DFD mediante un Diagrama de fluxo, si puidéramos coñecer cal é a súa tradución de cada símbolo a Wiring ( linguaxe de programación de Arduino ) poderíamos confeccionar programas neste entorno e resolver pequenos proxectos na aula taller conectando compoñentes discretos na placa. Na seguinte táboa ilústrase esta tradución:

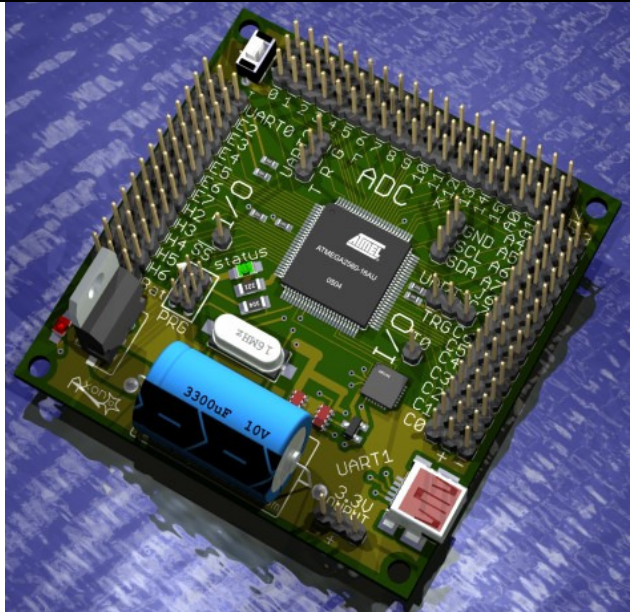
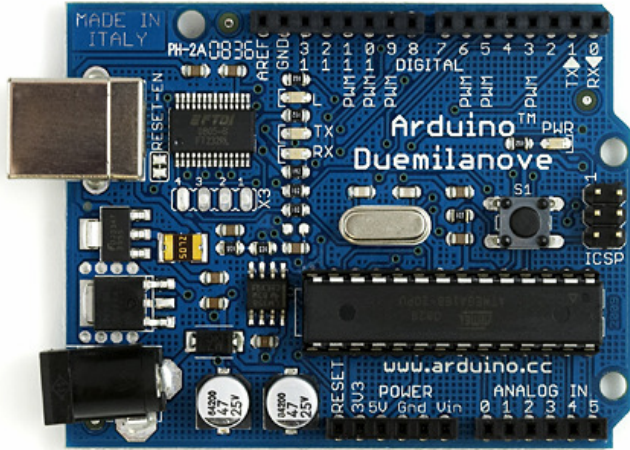

Simbología usada en DFD	Código usado con Arduino
	<pre>Serial.print("Hola Ourense");</pre>
	<pre>Serial.read(A);</pre>
	<pre>if (A &lt; 500) { resta=A-B; } else { suma=A+B; }</pre>
	<pre>A=B+1;</pre>
	<pre>for ( i=0; i&lt;10; i++) { Suma=A+1; }</pre>

**EXEMPLOS PROPOSTOS PARA ARDUINO:**

1. Partindo do parpadeo dun LED e xogando ca función delay():
  - Diseñar un SEMÁFORO. (Tres leds conectados ás respectivas saídas con R<sub>Limitadora</sub>)
  - Diseñar un CRUCE DE SEMÁFOROS cunha intersección (6 Leds). Incluir paso de peóns con pulsador.
  - Diseñar os intermitentes dun auto onde Arduino pode ser a centraliña.
  
2. Partindo dun ou varios bucle “if” e utilizando pulsadores ou finais de carreira:
  - Encendido dun LED a vontade cun interruptor ou pulsador.
  - Robot rastreador con sensores de obxectos.
  - Robot segueliñas. (con CNY70 e L293 )
  - Robot con sensor de infravermellos.
  - Porta de garaxe con apertura por proximidade.
  
3. Usando funcións, bucles e integrados varios:
  - Tacómetro. (Mide rpm dun motor)
  - Frenado/arrancado controlado de motores DC.
  - Steppers (Motores paso a paso).
  - Contadores de persoas para centro comercial.
  - Servomotores.
  
4. Usando código en xeral, Zigbee (módulo inalámbrico) e programación avanzada con Interrupcións pódese construír e deseñar de todo.
  - Robot manexado a distancia con joystic.
  - Vehículo autónomo sen condutor.
  - Etc....

**ANEXO II**

**Comparativa entra placas programables para robótica**

 <p>A photograph of the AXON programmable board, a green PCB with a large number of gold pins on the sides. It features an ATmega168 microcontroller, a blue 3300µF capacitor, and various other electronic components.</p>	<p><b><u>PLACA AXON (111 Euros+ IVA)</u></b></p> <ul style="list-style-type: none"> <li>16 entradas analógicas</li> <li>Conectables 29 Servos.</li> <li>Bus I2C</li> <li>55 I/O Totais</li> <li>18 interrupcións</li> <li>9 PWM</li> <li>64KB Flash, 4KB EEPROM, 8KB SRAM</li> <li>6 Timers (catro 16-bit, dous de 8-bit)</li> <li>Trae programado o bootloader</li> <li>Buses de potencia 1.5A</li> <li>Software gratuito</li> <li>100% open source</li> </ul>
 <p>A photograph of the Arduino Duemilanove board, a blue PCB with a USB Type-B port, a DC power jack, and a reset button. It features an ATmega168 microcontroller and various passive components.</p>	<p><b><u>Arduino / Arduclema (28Euros + IVA)</u></b></p> <ul style="list-style-type: none"> <li>Microcontrolador ATmega168</li> <li>Voltage de entrada (recomendado) 7-12 V</li> <li>Pines Digitales I/O. 14 (6 PWM)</li> <li>Pines de Entrada analógica 6</li> <li>DC Current per I/O Pin 40 mA</li> <li>Flash Memory 16 KB (2KB usados por el bootloader)</li> <li>SRAM 1 KB</li> <li>EEPROM 512 bytes</li> </ul>
 <p>A photograph of the Arduino MEGA board, a blue PCB with a USB Type-B port, a DC power jack, and a reset button. It features an ATmega1280 microcontroller and various passive components.</p>	<p><b><u>Arduino MEGA (60Euros+IVA)</u></b></p> <ul style="list-style-type: none"> <li>Microcontrolador: ATmega1280</li> <li>Tensión de entrada recomendada: 7-12V</li> <li>Límite de entrada: 6-20V</li> <li>Pines digitales: 54 (14 con PWM)</li> <li>Entradas analógicas: 16</li> <li>Corriente máxima por pin: 40 mA</li> <li>Corriente máxima para el pin 3.3V: 50 mA</li> <li>Memoria flash: 128 KB (4 KB usado por el bootloader)</li> <li>SRAM: 8 KB</li> <li>EEPROM: 4 KB</li> </ul>

Páxinas web:

[www.robotshop.com](http://www.robotshop.com) (Canadá) Placa AXOM

[www.libelium.com](http://www.libelium.com) (España) Arduino e Arduino MEGA

[www.ray-ie.com](http://www.ray-ie.com) (España) Arduclema e módulos Xbee.

[www.bricogeek.com](http://www.bricogeek.com) (Carballo, España) Arduino MEGA e Arduino/módulos varios IR.