Elementos básicos de la programación

En este texto se presentan los elementos mínimos básicos que deben considerarse en el momento de construir un programa, en particular para un sistema basado en un circuito microcontrolador.

Diseño de un sistema electrónico basado en un circuito microcontrolador.

El diseñar un sistema electrónico basado en un microcontrolador es una tarea interesante, productiva y divertida. Habitualmente esta labor se compone de dos etapas, el diseño de la circuitería (*hardware*) y el diseño de la programación (*software*).

Se considera a la circuitería como el conjunto de elementos físicos que configuran al sistema, incluye entre otras cosas el alambrado, los conectores, el gabinete, los elementos mecánicos y algunas cuestiones físicas de otra naturaleza.

El usuario del sistema interactúa con el sistema que diseñamos a través de los elementos de la circuitería. Oprime un botón o interruptor, observa luces que indican el estado del sistema, oye el pitido de una sirena que anuncia algún problema o defecto de funcionamiento, observa una serie de letreros o gráficas que indican instrucciones a seguir para emplear adecuadamente el artefacto construido.

La programación del sistema permite orquestar de manera adecuada los distintos recursos de que consta (circuitería, hardware) para desempeñar satisfactoriamente las funciones para las que fue construido.

La programación es la principal labor que se puede desempeñar en el diseño de un sistema computacional (basado en algún elemento de cómputo). El arreglo de los elementos físicos habitualmente está predeterminado por una serie de sugerencias o reglas. La programación está en buena medida orientada por diversas características de quien le desarrolla: conocimiento e interpretación del problema a resolver; dominio del lenguaje de programación que se emplea; formación en diversas áreas de la ingeniería, entre otras cuestiones.

Elementos básicos que se debieran considerar al afrontar las actividades de programación se resumen en lo siguiente:

+ Conocer el problema que se pretende resolver.
+ Trasladar los elementos del problema a las características del sistema en donde se desarrolla la solución.
+ Asociar los elementos conocidos (entradas) a mecanismos de programación (datos), así como relacionar soluciones (salidas) a órdenes del lenguaje de programación (variables).
+ Encontrar la secuencia de pasos que implementan la solución al problema, a esto se le denomina algoritmo de programación.
+ Describir en una manera simple el algoritmo encontrado, para evaluar el adecuado funcionamiento en todas las condiciones a las que se sujetará el sistema que se diseña.
+ Trasladar el algoritmo ensayado al código específico, de acuerdo al lenguaje de programación seleccionado.
+ Verificar el funcionamiento del sistema empleando la circuitería y programación diseñada. Habitualmente se ocupa alguna herramienta de simulación para estos fines.
+ De ser necesario afinar alguna cuestión, se repiten los pasos anteriores.

Ahora nos referiremos a cuestiones relacionadas con la programación.

Algoritmo.

En el ámbito de la computación se define *algoritmo* como la secuencia de pasos finita que permite la solución de un problema. Frecuentemente al algoritmo se le plantea como una receta o lista de pasos a seguir para llegar a la solución de un cierto problema. Para considerarse como un algoritmo válido, la secuencia de pasos debe poder completarse en alguna manera, dentro de un tiempo conocido, por eso se dice que es una secuencia finita de pasos.

Para la elaboración de un algoritmo es común emplear como medio de expresión al "pseudocódigo", el cual consiste en un listado de instrucciones establecidas en lenguaje común, sin que la descripción se apegue a algún lenguaje de programación específico.

A continuación se presenta un algoritmo empleado para realizar el cambio de una llanta una vez que se ha detectado que está desinflada:

```
1. Estacionar el automóvil y aplicar el freno de mano.
2. Apagar el automóvil.
3. Sacar la llanta de refacción y las herramientas necesarias
(gato, llaves, retenes).
4. Colocar retenes en cada una de la llantas, a excepción de
la llanta a cambiar.
```

5. Colocar el gato en la posición adecuada (en la posición más cercana a la llanta que se desea cambiar).
6. Aflojar los tornillos de la llanta a cambiar.
7. Elevar el gato la distancia suficiente para que la llanta a cambiar pueda ser liberada.
8. Quitar la llanta ponchada.
9. Colocar la llanta de refacción en la posición adecuada.
10. Colocar los tornillos en la llanta colocada, apretando con la mano.
11. Bajar el gato de modo que el automóvil repose en posición normal.
12. Apretar las tuercas de la llanta recién cambiada con la llave adecuada.
13. Retirar los retenes de las llantas.
14. Guardar la llanta de refacción, las herramientas y retenes.

Algunas cuestiones que se establecen en el algoritmo se dan por sobreentendidas, por ejemplo: elevar el gato lo suficiente para…; el automóvil repose en posición normal; etc.

En el desarrollo de programas de cómputo es común emplear elementos gráficos para describir los algoritmos, a continuación se plantea uno muy popular.

Diagrama de Flujo.

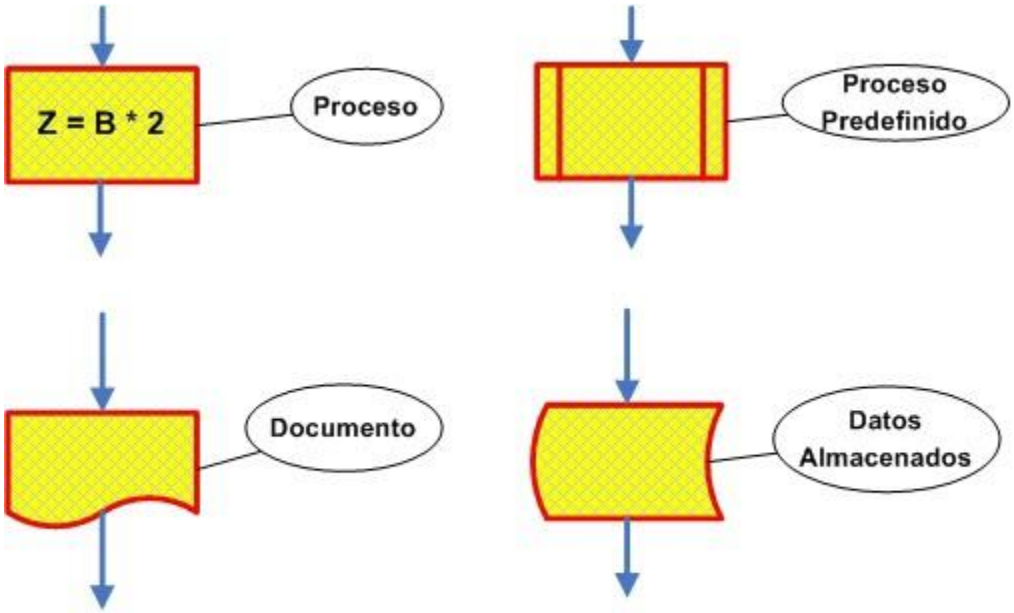
El diagrama de flujo es un método popular para expresar las acciones básicas que se efectúan a través de instrucciones en un programa de cómputo. Existen diversos símbolos para representar acciones, decisiones, entradas, salidas, procesos, conectores y otras cuestiones. Se presentan los principales elementos a continuación:

Proceso: Es un rectángulo que indica alguna acción, puede ser una operación aritmética, lógica, de introducción o expulsión de información, ajuste de valores, etcétera. Este símbolo tiene una entrada y una salida.

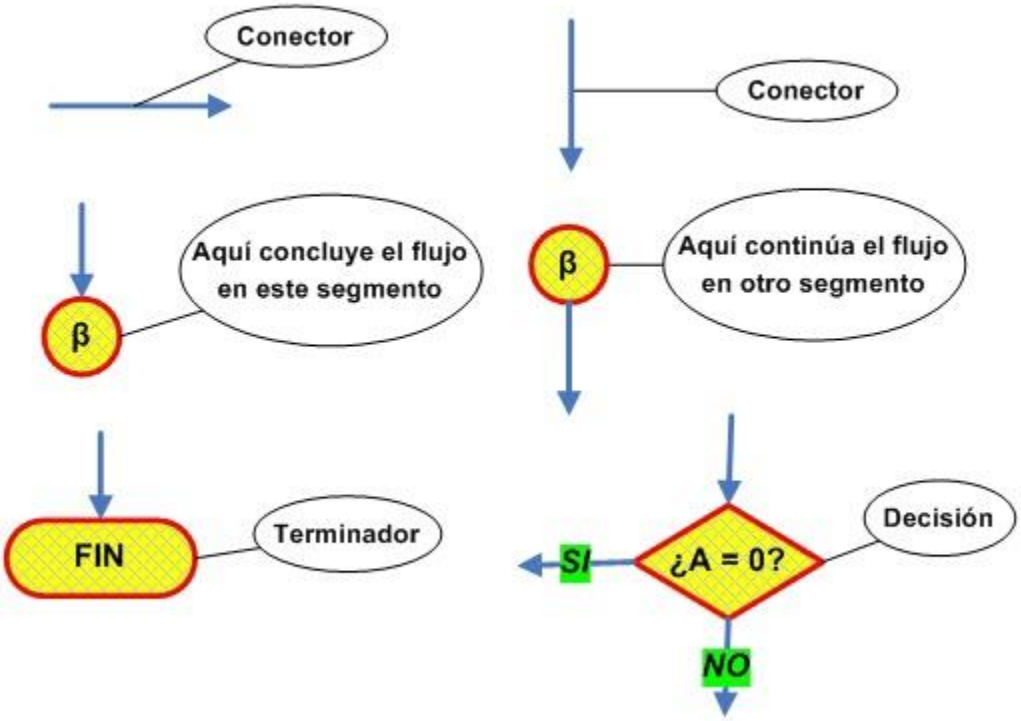
Proceso predefinido: Este rectángulo tiene líneas horizontales en los extremos. Identifica a una secuencia o serie de acciones, puede agrupar a varios procesos o bloques de acciones. Tiene una entrada y una salida. Identifica a estructuras conocidas como rutinas o subrutinas en diversos lenguajes de programación.

Documento: En algunos lenguajes indica la escritura o salida de información. Aunque no es frecuentemente empleada, puede identificar acciones específicas de escritura o impresión.

Datos almacenados: Esta forma establece el guardado de datos en algún medio específico, no es de uso frecuente.



Otros elementos comunes se muestran a continuación.



Conectores: Permiten el enlace entre los diversos símbolos del diagrama de flujo.

Conexiones: Círculos pequeños que facilitan enlazar o continuar segmentos de un mismo diagrama en distintas columnas.
Decisión: Es un rombo que indica la selección basada en un cálculo aritmético o lógico. Habitualmente consta de una entrada y dos salidas. Una de las salidas se identifica como la "verdadera", la cual se toma cuando es válida la pregunta o decisión, la otra salida es la "falsa", esta se selecciona cuando no se cumple la decisión.
Terminador: Este símbolo denota el final del algoritmo.

Estructuras de control de flujo

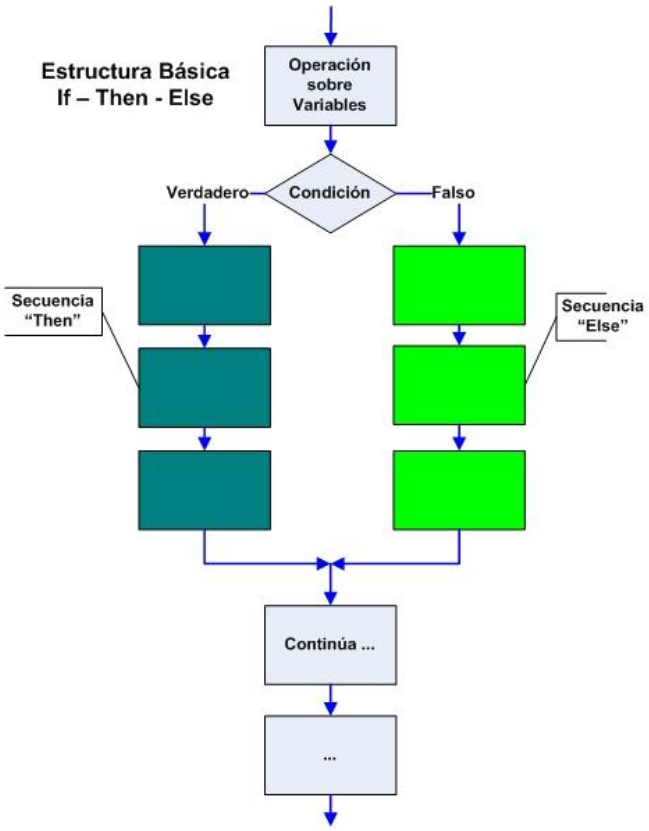
Las estructuras de *control de flujo* establecen la forma en la cual continúa la ejecución de un programa de cómputo, habitualmente en función de la evaluación de una condición.

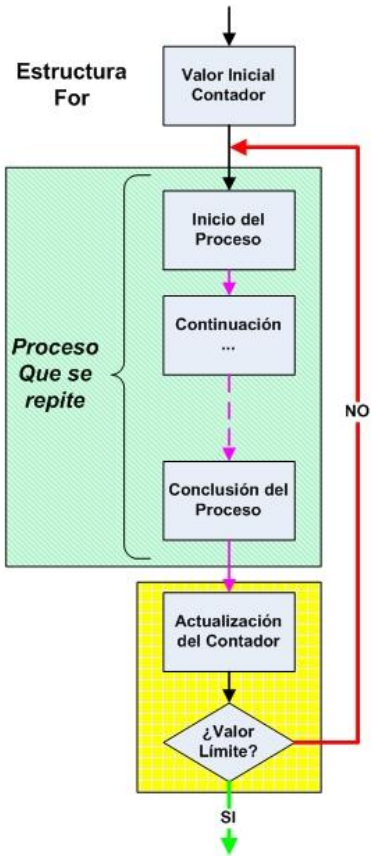
El control de flujo en la ejecución de un programa permite que en el código se planteen distintos caminos de ejecución, estos se ejecutan en función del tipo o valor de datos de entrada.

En el diseño de programas de cómputo se plantean condiciones, a partir de las cuales se toman decisiones. Las condiciones pueden ser valores de datos (igual a cinco, distinto a ocho), rangos de valores (mayor a diez, menor a veinte), condiciones (positivo o negativo), tipo de dato (par, impar) o alguna otra cuestión. Es importante que la condición evaluada se pueda distinguir en dos casos únicos, mutuamente excluyentes: la condición evaluada puede ser "*verdadera*" o "*falsa*". En función de cómo se puede estructurar la condición evaluada y el control del flujo de ejecución se pueden distinguir las siguientes formas básicas.

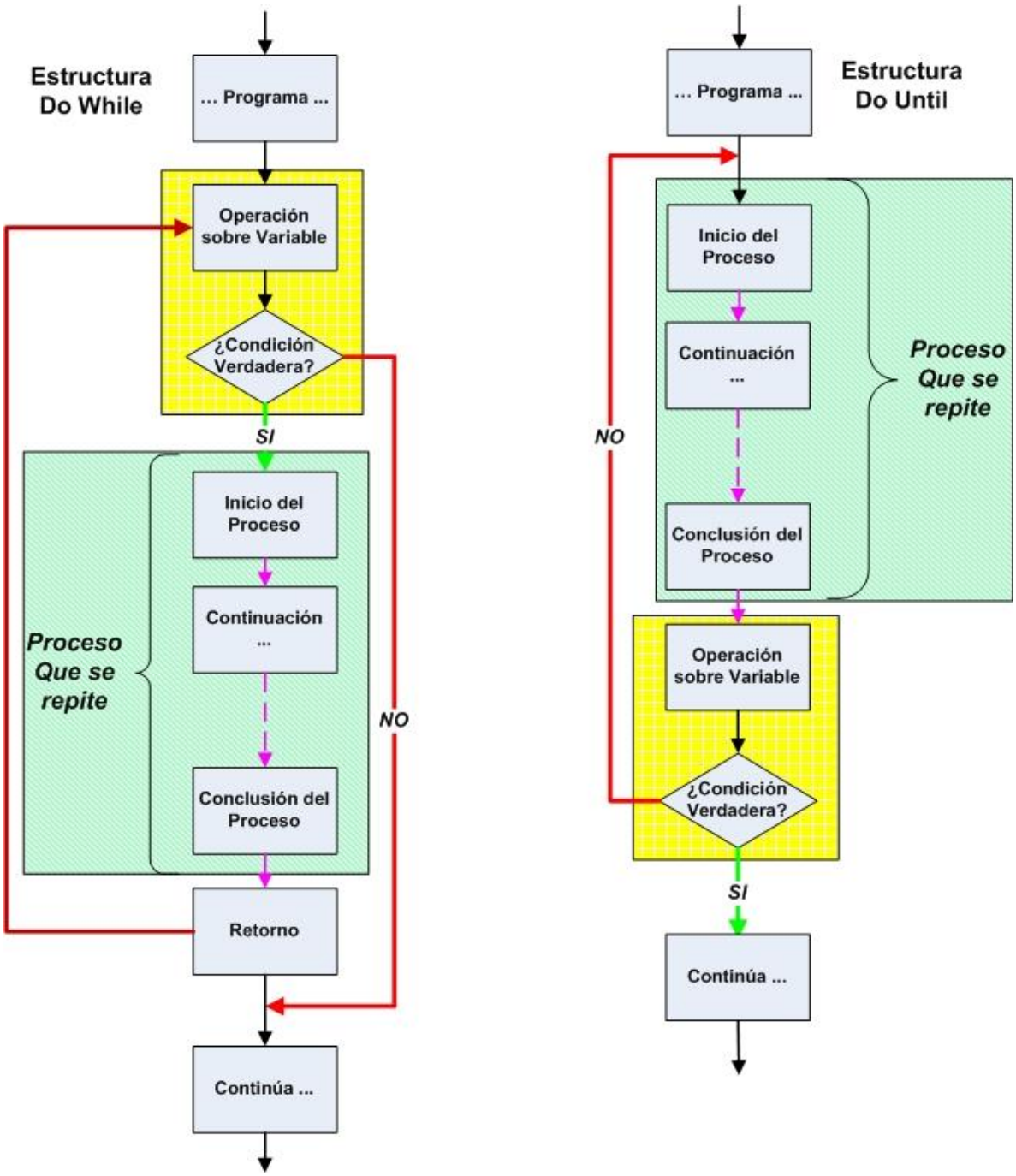
Estructura IF – Then – Else. En este esquema se toma una de dos trayectorias posibles en función del valor de una condición dada.

Estructura For. En esta estructura se repite una cierta parte de código, un segmento predeterminado. El número de veces que se repetirá el código ya es conocido y controlado comúnmente por un contador, el cual se inicia a un valor en particular. Posteriormente se actualiza el valor del contador (incrementa, decrementa) y se verifica si llegó a un valor preestablecido. Esta estructura permite realizar alguna acción un número de veces predefinido.





Estructura Do – While. En este caso se ejecuta una zona de código en función de la validez de cierta condición. En esta estructura primeramente se analiza el valor de la condición. Si la condición resultara *falsa*, no se ejecuta el segmento de código seleccionado. Si la condición es *verdadera* se ejecuta el segmento de código que hubiéramos propuesto. Una vez concluida la ejecución del segmento de código se vuelve a revisar el valor de la condición, repitiéndose la ejecución del segmento de código (si fuera *verdadera*) o saliéndose de esta estructura (si fuera *falsa*). En algunas ocasiones la condición es una variable de entrada o alguna variable que se altera durante la ejecución del segmento de código.



Estructura Do-Until. En esta estructura, a diferencia de la anterior, primeramente se ejecuta el segmento de código, al menos una vez. Cuando se concluye la ejecución del segmento de código se hace el análisis de la condición, repitiéndose la ejecución del segmento (si no se ha llegado o alcanzado la condición) o saliéndose de la estructura (si ya se llegó o alcanzó la condición).

Lenguaje de programación.

Para construir un programa que se ejecutará en una computadora o en un microcontrolador es necesario emplear un lenguaje que permita expresar las instrucciones y el algoritmo adecuado. Existen diversos lenguajes de programación, algunos de los más populares para circuitos microcontroladores son C, Basic, Pascal y Forth. El tipo de lenguaje que se emplee depende en buena medida de la experiencia de quien diseña el sistema, las facilidades que el lenguaje proporciona para el diseño de la aplicación, disponibilidad de soporte para el desarrollo del sistema y otras cuestiones.

Ensamblado y compilación.

En general el proceso de compilación o ensamblado es el traslado de información de una forma simbólica, reconocida por un humano, hacia una forma codificada, reconocida por una computadora. Se realiza una compilación cuando se traslada desde un lenguaje de alto nivel hacia código que ejecutará la computadora. Se realiza un proceso de ensamblado cuando se traslada información desde un lenguaje ensamblador (que se reconoce como un lenguaje de programación de bajo nivel) hacia el código máquina.