Capítulo 4

Estructuras Básicas y Técnicas para Representar Algoritmos

Estructuras Básicas

El concepto de Estructura

Una estructura se define como un esquema que nos permite representar de manera simplificada alguna idea y que bajo condiciones normales es constante. Ello significa que si hablamos de este concepto en esta parte del Libro significará que de alguna manera el pensamiento del ser humano, en lo que se refiere a los algoritmos, está enmarcado en algún tipo de Estructuras que no solo le permiten tener un medio mas simplificado y a la mano para expresar las ideas sino que además permite "restringir" un poco el horizonte de la Lógica Algorítmica.

Es pertinente, pues, hacer un breve paréntesis para explicar el porqué es importante "restringir" un poco el horizonte de la Lógica Algorítmica. Comencemos con dos breves preguntas

1. Con cuántos algoritmos las señoras de la casa pueden preparar los fríjoles..?
2. Cuántas personas ve usted cerca que lleven puesta una camisa y un pantalón exactamente igual al suyo..?

La primera pregunta se resuelve preguntándole a tres o cuatro señoras acerca de su forma de preparar los fríjoles. Tenga la seguridad de que todas van a tener una manera diferente (o sea un algoritmo diferente) para prepararlos si los vemos detalladamente pero lo que va a ser coincidente en todas también es que logran el mismo objetivo que es dejar listos los fríjoles para ser degustados. Ello nos va a demostrar, en primera instancia, que cada persona concibe algorítmicamente el mismo proceso de manera diferente pero que pueden llegar al mismo objetivo sin importar el camino que hayan escogido para lograrlo.

La segunda es todavía mas reveladora y la voy a hacer en otro sentido Sabe usted porqué ninguna o casi casi ninguna persona lleva puesta una camisa y un pantalón exactamente igual al suyo..? Pues sencillamente porque todas las personas están cumpliendo, en condiciones normales, con el objetivo de estar vestidos mas no exactamente de la misma forma.

Esa variabilidad en cuanto a la concepción de un determinado algoritmo es lo que llevó a pensar en que la parte técnica también podría llegar a ser igualmente variable o mas bien exageradamente variable. Qué pasaba si una persona concebía un algoritmo computacional en unas condiciones lógicas que prácticamente solo ella la entendiera..? Pues precisamente que el día que esa persona fuera despedida de la empresa o se fuera o falleciera, la empresa se vería en un verdadero y grande problema.

A nivel informal la variabilidad de ópticas en cuanto a la concepción del mundo es lo que le ha permitido a éste avanzar y es de allí que se ha podido extractar tecnologías, modas, teorías y muchos avances del mundo moderno pero a nivel técnico si resulta ser muy importante que la lógica para desarrollar un algoritmo computacional sea tan clara que y tan “standard” (si se puede decir así) que un programa desarrollado por una persona sea fácilmente entendible por cualquier otra, dado que haciendo uso de la lógica propia de cada uno podemos llegar a encontrarnos con programas tan confusos que solo llegarían a ser entendibles por su creador.

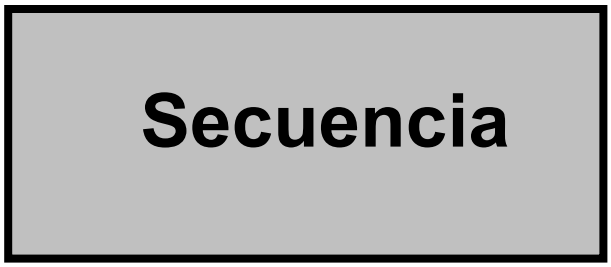
Esa es la razón fundamental por la cual se buscó “uniformar” la lógica para desarrollar algoritmos computacionales y poder llegar a unas estructuras básicas sobre las cuales se pueda decir que está fundamentada dicha Lógica.

Consideraciones Algorítmicas sobre el pensamiento humano

Precisamente y luego de analizar desde muchos ángulos el pensamiento humano y teniendo en cuenta los conceptos de algoritmo informal y algoritmo computacional se llegó a la conclusión de que dicho pensamiento se mueve entre tres estructuras básicas:

Cuando usted está planeando ir este fin de semana a pasear con la familia lo que en su mente se va dibujando poco a poco es una secuencia de acciones a realizar y que le permitan pasar un fin de semana bien bueno. Cuando usted tiene que pensar que debe ir hasta el paradero de buses a tomar el transporte lo que va organizando en su mente es una secuencia de acciones que le

permitan acercarse al paradero, esperar el bus correcto y tomarlo para irse para su casa. Pues bien esa es la primera estructura sobre la cual se mueve el pensamiento humano y es la estructura de



Permanentemente usted está inmerso en esta estructura y generalmente usted primero planea cada secuencia de acciones (consciente o inconscientemente) antes de ejecutarlas. Cada una de las cosas que hacemos diariamente no son mas que secuencias de acciones que hemos planeado para poder cumplir con nuestros objetivos en la sociedad.

Usted ha llegado al paradero de buses, ve como pasan y pasan buses pero ninguno tiene la ruta que necesita porque usted vive en ese barrio para el cual hay un transporte muy deficiente. Por cada bus que pasa usted le mira la ruta y al ver que no es, espera el siguiente bus y así sucesivamente hasta que ve llegar al bus que usted necesita. Usted está planeando el fin de semana pero no sabe si pasar el domingo en el balneario que queda a 1 hora de la ciudad en donde vive o aprovechar e ir hasta la finca de aquel tío que hace mucho tiempo no visita y que queda también a una hora.

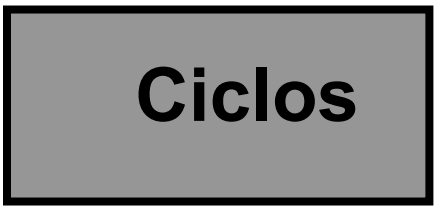
En cada alternativa encuentra ventajas y desventajas y usted sabe que visitar al tío es bueno porque hace mucho tiempo no lo vé pero también sabe que la finca del tío no tiene piscina y el balneario sí y que le gustaría ver a su familia divertirse en ella. Usted va a ver las noticias en algún noticiero de las 9:30 de la noche pero aún no sabe en qué noticiero verlas pues a esa hora presentan un noticiero diferente en cada uno de los canales de televisión. Situaciones muy normales diariamente usted ha vivido.

Precisamente usted está adportas de conocer la segunda estructura sobre la cual se basa el pensamiento (o razonamiento) humano. Esta es la estructura de

Gracias a la cual usted puede escoger lo que para usted sea la mejor alternativa de entre varias y hago hincapié en esto porque cuando usted tiene (como erradamente dicen los periodistas) una

sola alternativa pues sencillamente no tiene alternativa y no hay caminos para escoger. La decisión se da siempre que usted tenga que escoger de entre, por lo menos, dos caminos lógicos.

Usted acostumbra todos los días a ver el noticiero de las 9:30 de la noche, acostumbra a ir al trabajo a la misma hora y a esperar el bus en el mismo paradero, acostumbra saludar de la misma forma a su esposa y acostumbra dormir en el mismo lado y en la misma posición. Usted sencillamente vive practicando la tercer estructura y son los



Que no es mas que la estructura que nos permite repetir una o varias acciones una cantidad definida de veces. Todos los días usted almuerza en su casa según lo cual usted estará en el ciclo de ir a almorzar a su casa siempre pero en pleno almuerzo el hecho de que usted lleve muchas veces la cuchara del plato a su boca representa que usted estará haciendo lo mismo mientras en el plato exista todavía algo mas para comer. Puede notar que permanentemente (e infortunadamente) estamos también realizando tareas cíclicas. Cuando nos aferramos mucho a estos Ciclos de vida es cuando la vida se nos vuelve tremendamente monótona.

Por ahora lo que importa es que usted tenga claro que todo lo que usted hace, sin importar que sea, cualquier acción o conjunto de acciones que usted haga siempre estarán enmarcadas en estas tres estructuras

1. Secuencias de acciones
2. Decisión de acción
3. Ciclos de acciones

También conviene que sepa que tomar una decisión depende de una determinada condición y que repetir un conjunto de acciones depende de que se cumpla o se deje de cumplir igualmente una condición.

Estructuras Básicas expresadas Técnicamente

Precisamente y con el ánimo de facilitar unos patrones técnicos que permitan describir las ideas lógicas de una manera uniforme se han desarrollado unos esquemas que nos van a permitir escribir las estructuras mencionadas anteriormente.

Las secuencias de órdenes

Para escribir una secuencia de ordenes o acciones todo lo que tiene que hacer es colocar una nueva orden o una nueva acción después de la última que haya colocado. De esta manera se entiende la secuencialidad y la ordinalidad en la ejecución de esas acciones.

Vamos a desarrollar un algoritmo que nos permita asomarnos a la ventana, pero vamos a asumir que la ventana a donde nos queremos asomar ya está abierta y que no estamos muy distantes de la ventana. Entonces diríamos

Algoritmo para asomarnos a la ventana
Inicio
    Ubicar la ventana por la que nos queremos asomar
    Levantarnos del lugar en donde estemos sentados
    Avanzar hacia la ventana
    Llegar hasta tener la ventana muy muy cerquita
    Asomarnos por la ventana
Fin

Tal vez usted puede notar que el enunciado de este ejercicio tiene unas condiciones que parecen inoficiosas. La razón de la presencia de estas condiciones es que, solo por el ejemplo, no quería que intervinieran otro tipo de estructuras.

En el ejemplo dado usted puede ver que cada acción está antes de una y después de otra (excepto por supuesto la primera y la última). También puede notar que para que este algoritmo nos permita asomarnos a la ventana todo lo que tenemos que hacer es realizar cada acción en el orden en que están planteadas y sencillamente realizar una a la vez. Eso nos va a permitir lograr el objetivo propuesto.

Si queremos realizar el algoritmo para colocarnos una camisa (asumimos que la camisa está en nuestro ropero doblada y abrochada) entonces

Algoritmo para colocarnos una camisa
Inicio
    Dirigirnos a nuestro ropero
    Abrir el ropero
    Tomar una camisa
    Desabrocharla
    Abrir la camisa
    Meter un brazo por una de sus mangas
    Meter el otro brazo por la otra de sus mangas
    Ajustar la camisa al tronco
    Abotonarla (botón a botón)
Fin

Al igual que en el ejemplo anterior todo lo que tenemos que hacer es ejecutar cada acción en el orden indicado y hacerlo paso a paso y entonces podremos lograr el objetivo de colocarnos la camisa.

Puede usted notar que para utilizar la estructura de secuencia (que a veces parece ser tan obvia) todo lo que tenemos que hacer es ir colocando una acción tras otra y, por supuesto, ser muy racionales en el orden de dichas acciones porque estoy seguro que, hasta el momento, usted ha podido notar que en cuestión de algoritmos El orden de los factores sí altera el resultado.

Las Decisiones

Siempre que tenemos que tomar una decisión o, mas bien, siempre que tengamos que utilizar la estructura de Decisiones vamos a depender de una condición. La condición es la que nos permite que podamos decidir cuál es el camino lógico correcto a tomar.

Vamos a desarrollar el mismo algoritmo de asomarnos a la ventana pero esta vez no le vamos a colocar las condiciones de que estamos cerca de la ventana y de que ésta está abierta. Para ello vamos a incorporar una líneas de decisión que nos permitan tener un algoritmo mas genérico y que nos permita lograr mejor el objetivo, así :

```
Algoritmo para asomarnos a la ventana
Inicio
        Ubicar la ventana por la que nos queremos asomar
        Si estamos sentados
                Levantarnos del lugar en donde estemos sentados
                Orientarnos hacia la ventana
        Sino
                Orientarnos hacia la ventana
        Avanzar hacia la ventana
        Llegar hasta tener la ventana muy muy cerquita
        Si esta cerrada
                Abrirla
        Asomarnos por la ventana
Fin
```

Ya puede usted notar que nuestro algoritmo ha cambiado un poco y por lo tanto ahora tiene unas condiciones que le permiten ser una secuencia de acciones mas racional. En estas condiciones el algoritmo se convierte en algo mas depurado y mucho mas aproximado a la realidad. Note usted varias cosas en este algoritmo

1. Las palabras Si que aparecen son exclusivamente condicionales y no afirmativas como pudiera pensarse en algunos casos
2. Después de cada Si condicional va una condición que es la que permite que se haga una cosa u otra. La condición regula las acciones que vienen después y que dependen del Si condicional inicial. En la decisión

```
        Si estamos sentados
                Levantarnos del lugar en donde estemos sentados
                Orientarnos hacia la ventana
        Sino
```

Orientarnos hacia la ventana

Notamos que estar sentados es la condición de la cual depende si hacemos las dos acciones

Levantarnos del lugar en donde estemos sentados
Orientarnos hacia la ventana

O si solo hacemos la acción

Orientarnos hacia la ventana

3. Puede usted notar que una decisión completa involucra

Una pregunta que evalúa una condición
Un conjunto de acciones a realizar en caso de que la condición sea verdadera
Un conjunto de acciones a realizar en caso de que la condición sea falsa

Esta última parte, dentro del numeral 3, es razón de ser de la existencia de la acción Sino.

4. No siempre que exista un condicional *Si* debe existir un *Sino* asociado a él. Siempre que exista un *Sino* es porque está asociado a un *Si* condicional determinado. Tal es el caso de la Decisión

Si esta cerrada
    Abrirla

En donde si la ventana está abierta pues no hay que hacer mas que asomarse por ella pero si está cerrada debemos primero abrirla para poder asomarnos por ella.

Retomando el segundo ejemplo, y sabiendo que contamos con una estructura para mejorar los algoritmos, podremos adecuarlo de manera que el algoritmo para colocarnos una camisa quede de la siguiente forma

Algoritmo para colocarnos una camisa
Inicio
    Dirigirnos a nuestro ropero
    Si esta cerrado
        Abrirlo
    Tomar una camisa
    Si está abrochada
        Desabrocharla
        Abrir la camisa
    Si está doblada
        Desdoblarla
    Meter un brazo por una de sus mangas
    Meter el otro brazo por la otra de sus mangas
    Ajustar la camisa al tronco

```
        Si es una camisa de botones
                Abotonarla (botón a botón)
                Ajustarla al cuerpo
        Sino
                ajustarla de manera que quede bien puesta
Fin
```

Claramente aquí se puede notar una utilización alta de Condicionales Si que no tienen mucha necesidad de tener un Sino por las razones lógicas del mismo algoritmo. Es obvio que usted podrá tener muchos "reparos" a este algoritmo porque en algún sentido alguno o algunos de los pasos aquí consignados no coinciden con su lógica, pero tenga en cuenta que todos los algoritmos planteados en este libro son solo una idea del autor y que si su idea es acertada, es decir logra los mismos objetivos, así el algoritmo sea diferente estará completamente correcto.

Sé que han de existir muchas diferencias de concepción sobre todo en cuanto a este tipo de algoritmos informales pero lo importante es que usted se vaya acostumbrando a una filosofía propia de los algoritmos para expresar cualquier idea.

Los Ciclos

Vamos a suponer para ampliar nuestros ejemplos que usted es un supervisor de una fábrica y que cada media hora, a lo largo de todo el día, debe estar vigilando determinada acción a través de una ventana. El algoritmo para cumplir su objetivo que es el de Vigilar (como supervisor de la fábrica) parte de una unidad muy sencilla y es Asomarse por una ventana. En palabras sencillas usted tendrá que asomarse por una ventana mientras no termine el día cada media hora y durante el tiempo que usted no esté asomado lo que tiene que hacer es seguir en su puesto de trabajo. De esta forma, y partiendo de lo que ya tenemos, usted podrá estructurar un algoritmo mas o menos de la siguiente manera

```
Algoritmo para Vigilar desde una ventana
Inicio
        Llegar puntual a la hora de inicio de la jornada laboral
        Ubicarnos en nuestro escritorio
        Mientras no sea el fin del día
                Ubicar la ventana por la que nos queremos asomar
                Si estamos sentados
                        Levantarnos del lugar en donde estemos sentados
                        Orientarnos hacia la ventana
                Sino
                        Orientarnos hacia la ventana
                Avanzar hacia la ventana
                Llegar hasta tener la ventana muy muy cerquita
                Si esta cerrada
                        Abrirla
                Asomarnos por la ventana
                Regresar a nuestro escritorio
                Mientras no haya pasado Media Hora
                        Permanecer en nuestro escritorio
                Fin_Mientras
        Fin_Mientras
Fin
```

Varios factores nuevos entran en este algoritmo

1. La palabra Mientras establece en relación con una condición el inicio de un conjunto de acciones que se repiten precisamente Mientras esa condición lo permita
2. Todo Mientras (por efectos de clarificación del algoritmo) debe tener un finalizador que indique hasta donde llega el bloque de acciones que debemos repetir.
3. La indentación o lo que corrientemente se conoce como el "Sangrado" del texto es decir el hecho de que algunas acciones estén mas adentro de la hoja que otras, representa que existen bloques de acciones que tienen una característica.

   Las acciones contenidas entre el Inicio y el Fin indican que son las acciones que conforman el algoritmo en mención.

   Las acciones comprendidas entre Mientras no sea Fin del día y su correspondiente Fin_Mientras son el conjunto o bloque que se debe repetir (o iterar) precisamente mientras la condición sea Verdadera o sea Mientras no sea fin del día.

   La acción comprendida entre Mientras no haya pasado Media Hora y su correspondiente Fin_Mientras es la acción que se deberá realizar hasta cuando se complete media hora.

4. Cada ciclo de acciones que se inicie con Mientras deberá tener un Fin_Mientras asociado y a su vez cada Fin_Mientras deberá finalizar uno y solo un ciclo iniciado con Mientras.

Supongamos que usted es el inspector de calidad de un almacén de ropa y su trabajo consiste en medirse algunas de las camisas que están en los roperos del almacén para verificar su ajuste en cuanto a la talla. Entonces, mientras no termine su jornada de trabajo usted lo que hará será ir de ropero en ropero tomando una camisa y midiéndosela. De esta forma si partimos del algoritmo de colocarnos una camisa que ya tenemos entonces este nuevo objetivo puede cumplirse de la siguiente forma

```
Algoritmo Inspeccionar las camisas en un almacén de ropa
Inicio
        Llegar puntuales al inicio de la jornada laboral
        Mientras no sea fin de la jornada laboral
                Dirigirnos a un ropero
                Si esta cerrado
                        Abrirlo
                Tomar una camisa
                Si está abrochada
                        Desabrocharla
                        Abrir la camisa
                Si está doblada
                        Desdoblarla
                Meter un brazo por una de sus mangas
                Meter el otro brazo por la otra de sus mangas
                Ajustar la camisa al tronco
                Si es una camisa de botones
                        Abotonarla (botón a botón)
                        Ajustarla al cuerpo
                Sino
                        Ajustarla de manera que quede bien puesta
                Emitir el concepto de calidad sobre la camisa
        Fin_Mientras
```

Fin

Las apreciaciones acerca de este algoritmo coinciden en su mayoría con las apreciaciones acerca del algoritmo anterior (dentro de este mismo tema). Es de anotar que así como, por claridad, se utiliza un Fin_Mientras para indicar en donde termina el bloque de instrucciones que se deben operar es conveniente utilizar un Fin_Si para indicar hasta donde llega completamente una decisión y, en unión con la indentación de acciones, tener claridad en cuanto a los "bloques" de acciones que se formen dentro del algoritmo.

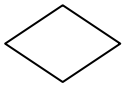
Estos algoritmos informales están expresados tal como desprevenidamente cualquier persona los expresaría y puede entonces suponer usted que la variabilidad de algoritmos que cumplan los mismos objetivos sería inmensa si no existieran unas técnicas uniformes para facilitar la expresión de estas ideas, particularmente en algoritmos computacionales.

Técnicas Para Representar Algoritmos

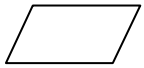
Diagramas de Flujo

Los Diagramas de Flujo parte de unos símbolos que nos permiten decirlo mismo que dijimos hace un momento en los algoritmos pero de una manera gráfica y, por supuesto, un poco mas entendible. Los siguientes son algunos de los símbolos (y el significado de ellos) que se han acordado utilizar dentro de los Diagramas de Flujo o Flujogramas son los siguientes

Un rectángulo representa un proceso que no es mas que una acción ó una orden a ejecutarse de manera clara y concreta. Un ejemplo típico de proceso es la asignación de un valor a una variable.

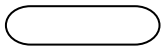


Este símbolo nos permite representar una Decisión. En su interior podemos escribir la condición de la cual depende la decisión y por sus extremos derecho (o izquierdo) e inferior se pueden colocar las salidas para los casos en que la condición sea Falsa o sea Verdadera.

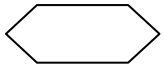


Este símbolo nos permite expresar un proceso de entrada o salida, teniendo en cuenta que una entrada en un algoritmo se concibe como el proceso a través del cual se recibe información y una salida es el proceso a través del cual se entrega información.

Este símbolo permite representar la escritura de un resultado o lo que técnicamente se conoce como una salida.

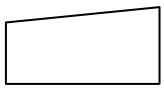


Este símbolo representa el Inicio ó el Fin de un Algoritmo. Todo lo que se tiene que hacer es escribir la palabra Inicio o Fin y ubicarlo apropiadamente dentro del Diagrama de Flujo.



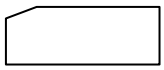
Este símbolo permite que coloquemos en él los parámetros de inicio de un ciclo cuando se ajusta a una de las formas establecidas por las normas de

programación. En el capítulo de Ciclos desglosaremos un poco mas esta definición.



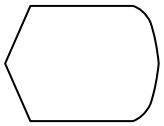
Este símbolo representa una entrada de datos utilizando el teclado del computador. Todo lo que tenemos que escribir en su interior es el nombre de la variable (o las variables) en donde queremos que se almacene el dato que entra por el teclado.

Estos símbolo se conocen como conectores lógicos. Nos permiten representar la continuación de un Diagrama de Flujo cuando éste es tan largo que no cabe en una sola hoja.

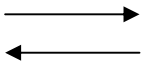


Este símbolo permite representar una lectura de datos. Representa una Tarjeta Perforada pues esta técnica fue establecida cuando aún se leían los datos a través de tarjetas perforadas. Actualmente este símbolo representa sencillamente una lectura.

Este símbolo genera una salida de datos. Representa una cinta perforada porque, al igual que el símbolo anterior, esta técnica fue establecida cuando aún se generaba la salida de datos a través de una tarjeta perforada. En la actualidad este símbolo representa sencillamente una salida o una escritura de datos.



Este símbolo representa una salida de datos pero escrita en la pantalla del computador. Es un símbolo un poco mas moderno para efectos de los diagramas de flujo.



Las flechas son los símbolos que nos van a permitir representar la forma de conexión entre los demás símbolos determinando igualmente el Flujo de ejecución o realización de acciones.

Estos símbolos (en unión con otros símbolos que para efectos de nuestro libro tal vez no haya necesidad de citar) fueron utilizados por mucho tiempo para representar gráficamente una idea o un algoritmo. Cómo se utiliza entonces esta simbología..? Tomemos el caso de los dos algoritmos que construimos mientras conocíamos las estructuras básicas.

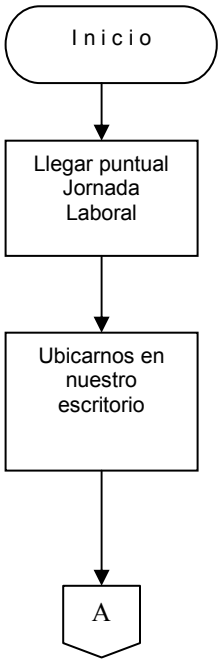
El enunciado final buscaba desarrollar una algoritmo que nos permitiera Vigilar una empresa desde una ventana asomándonos cada media hora por ella. El Algoritmo lo habíamos planteado como sigue a continuación

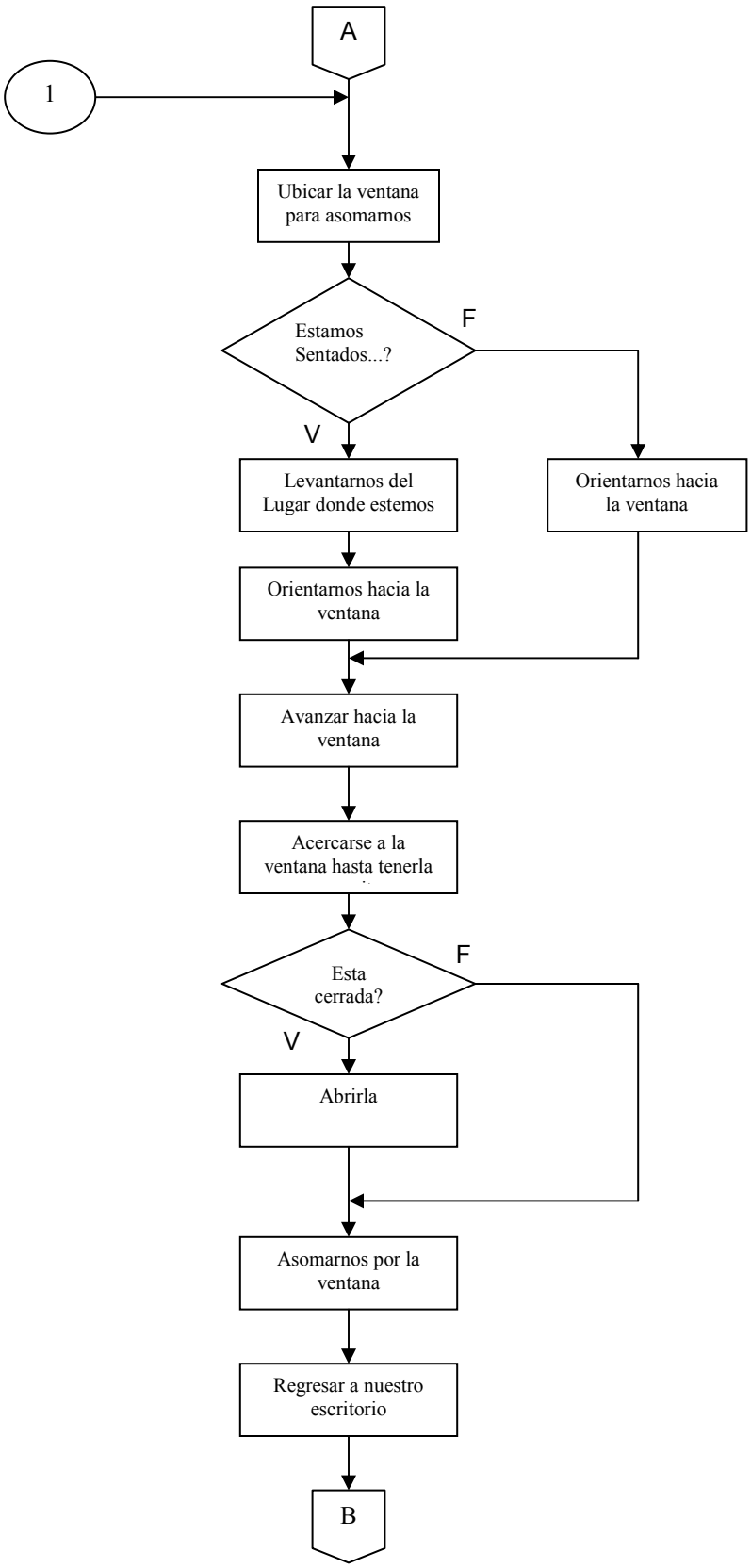
Algoritmo para Vigilar desde una ventana
Inicio
    Llegar puntual a la hora de inicio de la jornada laboral
    Ubicarnos en nuestro escritorio

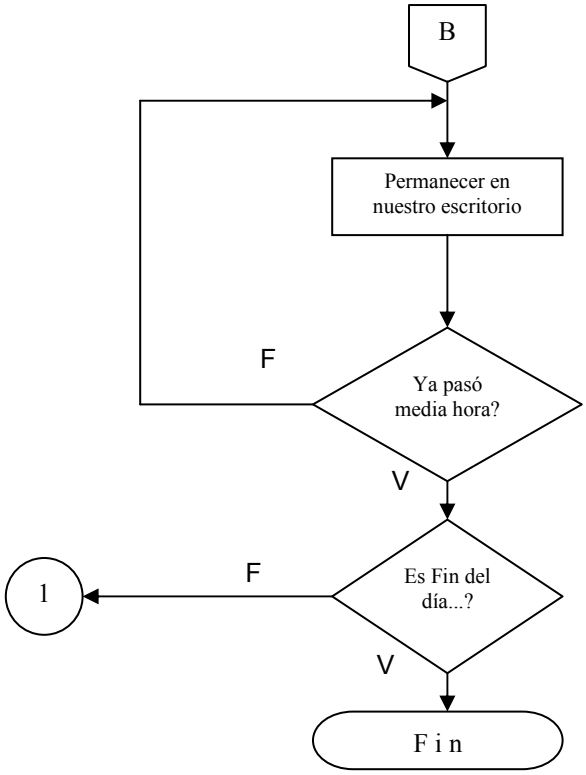
```
        Mientras no sea el fin del día
                Ubicar la ventana por la que nos queremos asomar
                Si estamos sentados
                        Levantarnos del lugar en donde estemos sentados
                        Orientarnos hacia la ventana
                Sino
                        Orientarnos hacia la ventana
                Avanzar hacia la ventana
                Llegar hasta tener la ventana muy muy cerquita
                Si esta cerrada
                        Abrirla
                Asomarnos por la ventana
                Regresar a nuestro escritorio
                Mientras no haya pasado Media Hora
                        Permanecer en nuestro escritorio
                Fin_Mientras
        Fin_Mientras
Fin
```

Si queremos llevarlo a la simbología de Diagramas de Flujo su equivalente sería el siguiente:

Diagrama De Flujo Para Vigilar Desde Una Ventana




A
1
Ubicar la ventana para asomarnos
Estamos Sentados...?
F
V
Levantarnos del Lugar donde estemos
Orientarnos hacia la ventana
Orientarnos hacia la ventana
Avanzar hacia la ventana
Acercarse a la ventana hasta tenerla
Esta cerrada?
F
V
Abrirla
Asomarnos por la ventana
Regresar a nuestro escritorio
B



Cabe destacar algunos detalles significativos en este Diagrama de Flujo:

1. Toda decisión, como es obvio, tiene dos caminos: Un camino nos lleva a la acción o las acciones a realizar en el caso de que la respuesta a la pregunta sea Verdadera y el otro camino es el que nos dice que debemos hacer en caso de que la respuesta a la pregunta sea Falsa.
2. Lo que en el algoritmo eran unos ciclos, en el diagrama se cambiaron por unas decisiones en donde uno de los caminos se devuelve (instrucciones atrás obviamente). Al realizar un seguimiento de este Diagrama de Flujo usted notará que se podrá devolver tantas veces como lo permita la condición de la Decisión que queda al final y que solo se va a salir de eses ciclo cuando la condición sea Verdadera o sea que el ciclo se mantiene mientras la condición sea Falsa lo cual concuerda con la teoría de los ciclos.
3. En la última decisión, el camino Falso nos lleva a una burbuja que tiene un número 1 adentro. Número que también está al principio del diagrama pero con la flecha en el otro sentido (es decir, no saliendo del diagrama sino entrando a él). Se utiliza esta notación solo para simplificar un poco el Diagrama de Flujo.
4. Con el Diagrama de Flujo usted puede ver un gráfico de la solución y con ello hacerse una idea clara de la secuencia de pasos que necesitaría para alcanzar el objetivo.
5. Siempre que vaya a desarrollar un Diagrama de Flujo trate de ser muy organizado y muy estético, pues no se olvide que si vamos a representar un algoritmo computacional (en donde se busca que el computador logre un objetivo por nosotros) al momento de la transcripción será muy importante el orden que usted haya tenido en la utilización de esta técnica.
6. Cuando diseñe un ciclo, no se olvide verificar que, lógicamente, la decisión por la cual reemplace el ciclo al momento de diseñar su diagrama de flujo tenga el mismo comportamiento

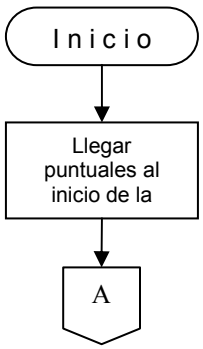
es decir permitan que bajo las mismas condiciones una acción o un conjunto de acciones se repitan una cantidad finita de veces.

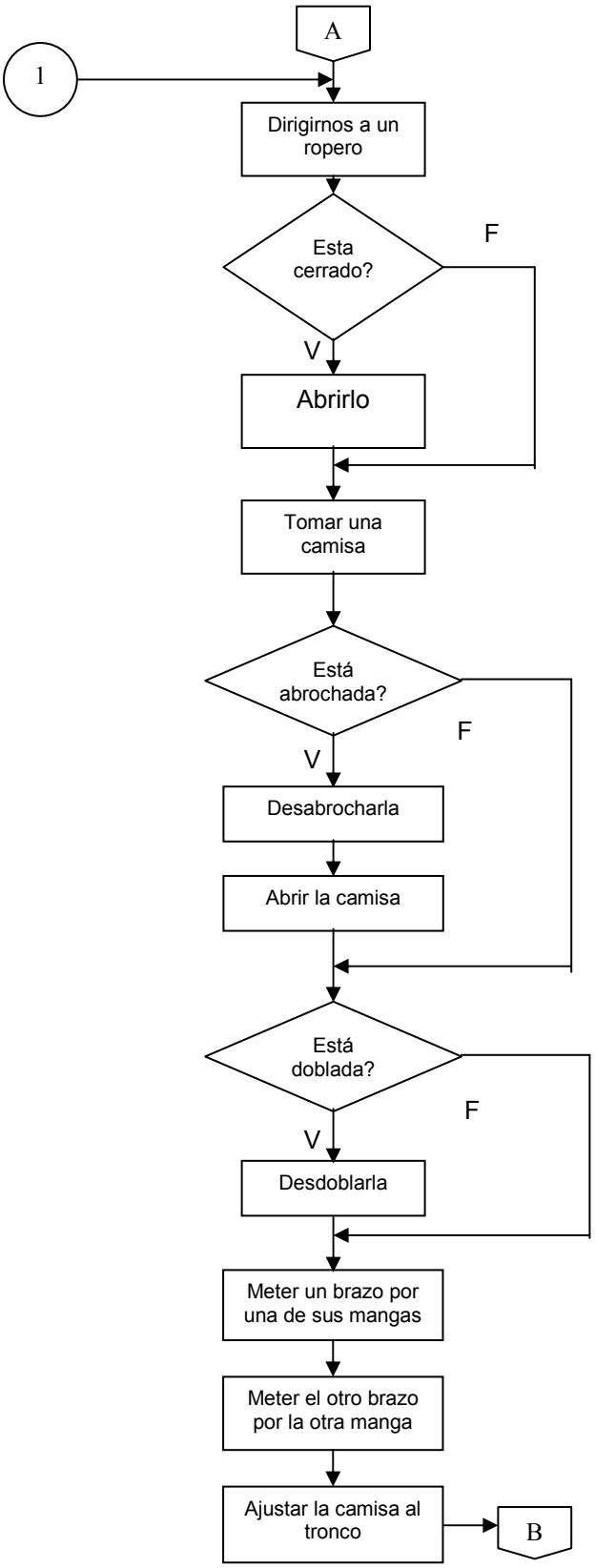
7. Si el algoritmo que usted tiene para lograr este mismo objetivo es diferente, tenga presenta que el Diagrama de Flujo también va a ser diferente ya que éste es un reflejo gráfico de aquel.
8. Es muy importante que sepa que el solo hecho de cambiar la llegada de una determinada flecha, cambia completamente el algoritmo. Puede usted notar que la utilización de los símbolos resulta ser una tarea muy simplificada, pero lo que si es delicado es la colocación de las flechas ya que ellas son las que representan el sentido con que se va a "mover" el flujo de nuestro lógica.

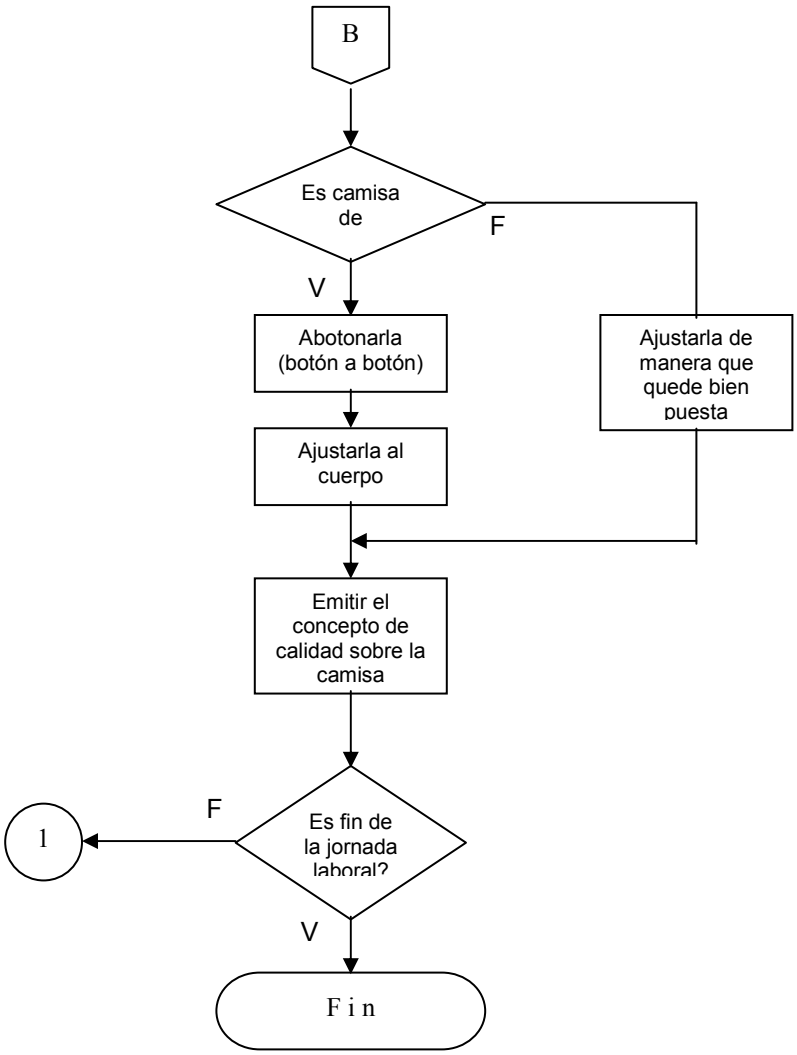
Para clarificar aún mas lo que hasta ahora hemos dicho, vamos a diseñar el diagrama de flujo del Algoritmo para inspeccionar las camisas en un almacén de ropa. Para ello, y tal como lo hicimos en el algoritmo anterior, partamos de la solución final que dimos a este algoritmo en donde involucrábamos secuencias de acciones, decisiones y ciclos-

```
Algoritmo para Inspeccionar las camisas en un almacén de ropa
Inicio
        Llegar puntuales al inicio de la jornada laboral
        Mientras no sea fin de la jornada laboral
                Dirigirnos a un ropero
                Si esta cerrado
                        Abrirlo
                Tomar una camisa
                Si está abrochada
                        Desabrocharla
                        Abrir la camisa
                Si está doblada
                        Desdoblarla
                Meter un brazo por una de sus mangas
                Meter el otro brazo por la otra de sus mangas
                Ajustar la camisa al tronco
                Si es una camisa de botones
                        Abotonarla (botón a botón)
                        Ajustarla al cuerpo
                Sino
                        Ajustarla de manera que quede bien puesta
                Emitir el concepto de calidad sobre la camisa
        Fin_Mientras
Fin
```

Llevado a la simbología de un Diagrama de Flujo, su equivalente sería el siguiente




A
1
Dirigirnos a un ropero
Esta cerrado?
F
V
Abrirlo
Tomar una camisa
Está abrochada?
F
V
Desabrocharla
Abrir la camisa
Está doblada?
F
V
Desdoblarla
Meter un brazo por una de sus mangas
Meter el otro brazo por la otra manga
Ajustar la camisa al tronco
B



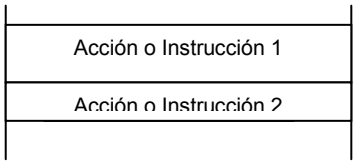
Estoy seguro que usted no estará de acuerdo conmigo en algunos de los pasos o acciones de este algoritmo (o del anterior) pero no se preocupe. Cada uno de nosotros actúa bajo una lógica propia pero enmarcado dentro de unas normas que son generales a todos. Si usted no está de acuerdo con los Diagramas expuestos hasta aquí será muy normal pues eso le demuestra que su algoritmo puede ser diferente al mío y diferente al de cualquier otra persona sin embargo estoy seguro que tanto su algoritmo como el mío lograrán el objetivo en ambos casos.

Es muy importante que sepa que, si a lo largo de este libro, usted tiene una idea diferente de cada algoritmo eso estará dentro de lo normal lo importante por ahora es que cada uno logre su objetivo. Diagrame su propia idea de estos dos algoritmos y verá, casi con toda seguridad, que su diagrama es completamente diferente a los expuestos en este libro. No olvide que en este libro aparecen solo las soluciones de una persona. El hecho de que sus soluciones no coincidan con las mías no necesariamente quiere decir que su solución esté mal.

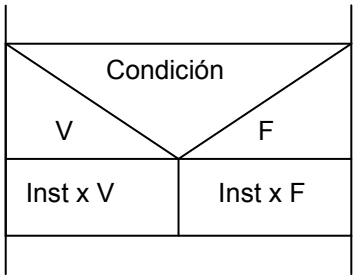
Recurra siempre a la prueba de escritorio antes de dudar de sus soluciones y además verifique las soluciones de este libro valiéndose de la misma herramienta.

Diagramas Rectangulares Estructurados

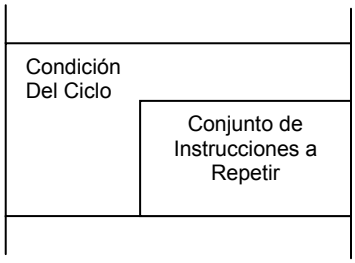
Una de las dificultades de los Diagramas de Flujo radica en que así como brinda la posibilidad de representar gráficamente el flujo de una idea o el "recorrido" de la solución a un problema también abre el espacio para que un programador desordenado ponga flechas de flujo a diestra y siniestra y finalmente obtenga una representación mas compleja que la idea misma. Precisamente la técnica de Diagramas Rectangulares Estructurados nos permite tener unas herramientas gráficas para representar la solución a un problema con la ventaja de que no brinda la posibilidad de que seamos desordenados en nuestra concepción. Gráficamente se basa en representar todo el algoritmo dentro del marco de un rectángulo y a diferencia de la técnica anterior, la DRE se mueve básicamente con la utilización de tres símbolos que corresponden a cada una de las estructuras básicas de la lógica de programación. Estas representaciones son las siguientes:



Para representar secuencias de instrucciones todo lo que tenemos que hacer es colocar cada instrucción en una línea "enmarcada".



Para representar una decisión se utiliza este símbolo en donde por el lado izquierdo podemos colocar las acciones o Instrucciones que correspondería ejecutar en el caso de que la condición fuera Verdadera y por el lado derecho colocaríamos las acciones o instrucciones a ejecutar cuando la condición fuera Falsa.



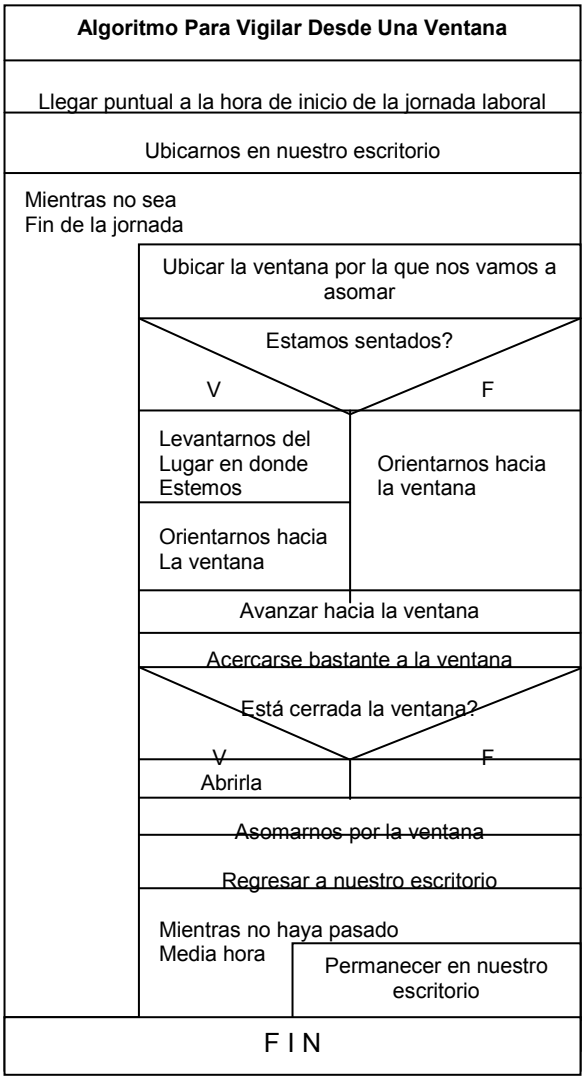
Para representar un ciclo sencillamente en la esquina superior izquierda del bloque correspondiente colocamos la condición y dentro del bloque colocamos las instrucciones o acciones que se debe repetir y que a su vez, por supuesto, dependen de la condición.

Pero definitivamente la utilización efectiva de esta técnica de representación se ve con un ejemplo. Vamos a estructurar en Diagramación Rectangular Estructurada los dos algoritmos de ejemplo que ya representamos en la Técnica de Diagrama de Flujo. Para ello volvamos a recordar el primer enunciado y su correspondiente solución a nivel Algorítmica. Se trataba de desarrollar un algoritmo que nos permitiera cada media hora durante la jornada de trabajo laboral vigilar desde una ventana. El algoritmo que fue planteado como solución final fue el siguiente:

Algoritmo para Vigilar desde una ventana
Inicio
    Llegar puntual a la hora de inicio de la jornada laboral
    Ubicarnos en nuestro escritorio
    Mientras no sea el fin del día

Ubicar la ventana por la que nos queremos asomar
Si estamos sentados
Levantarnos del lugar en donde estemos sentados
Orientarnos hacia la ventana
Sino
Orientarnos hacia la ventana
Avanzar hacia la ventana
Llegar hasta tener la ventana muy muy cerquita
Si esta cerrada
Abrirla
Asomarnos por la ventana
Regresar a nuestro escritorio
Mientras no haya pasado Media Hora
Permanecer en nuestro escritorio
Fin_Mientras
Fin_Mientras
Fin

Ahora bien, llevado este algoritmo a nivel de Diagramación Rectangular Estructurada el resultado sería el siguiente



Es imperante hacer algunas precisiones acerca de esta diagrama:

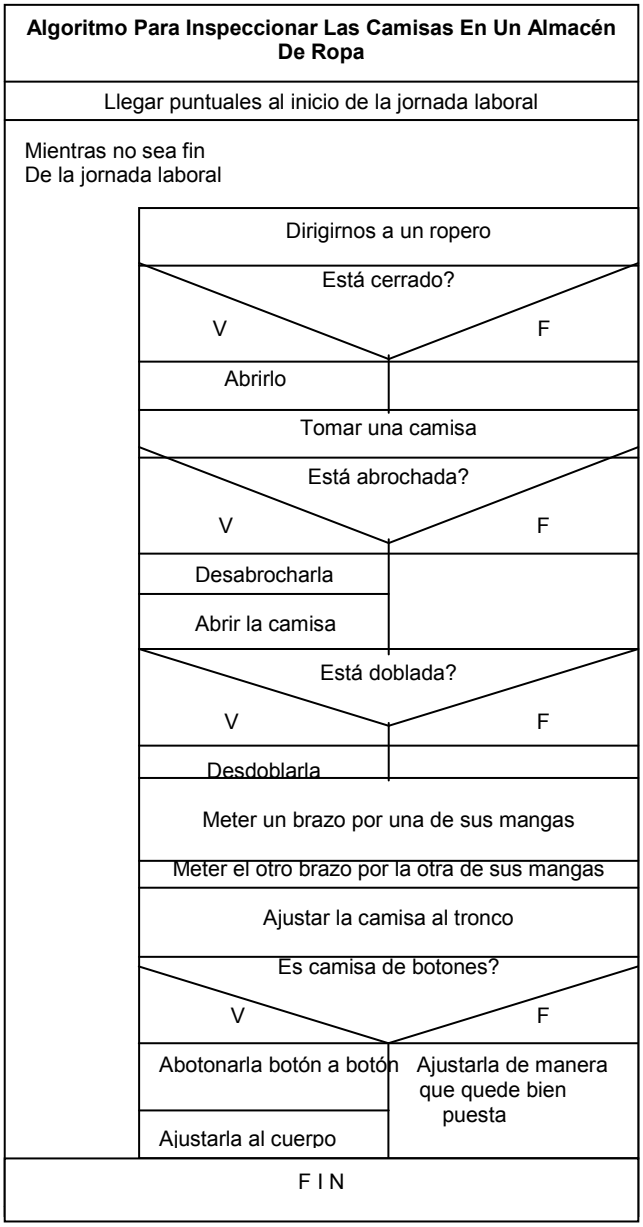
a. Puede usted notar que la correspondencia entre nuestra idea y su representación (bajo esta técnica) es mucho mas exacta que en el caso del Diagrama de Flujo en donde tuvimos que hacer algunos pequeños cambios lógicos para que el diagrama correspondiera a la solución planteada.
b. La técnica de diagramación rectangular estructurada obliga a ser mucho mas ordenado y no da ningún espacio para que nuestro algoritmo sea inentendible dado que las estructuras son relativamente rígidas.
c. Para la utilización de esta técnica solo tenemos que conocer tres símbolos y con ellos representamos todo lo que queramos dado que nuestra lógica se basa en esas tres estructuras.
d. Enmarcar nuestra idea en un rectángulo nos brinda una concepción mas concreta de la solución planteada.
e. Realizar una prueba de escritorio con un diagrama basado en esta técnica se reduce a seguir la secuencia de instrucciones y (al igual que con los diagramas de flujo) a realizar una a una y tal como están allí las instrucciones o acciones, las decisiones y la revisión de las condiciones de los ciclos.

Tomemos el segundo algoritmo y realicemos su correspondiente Diagrama Rectangular Estructurado. El enunciado buscaba diseñar un algoritmo para inspeccionar la calidad de las camisas en un almacén de ropa. La solución algorítmica que se planteó fue la siguiente

Algoritmo para Inspeccionar las camisas en un almacén de ropa

```
Inicio
        Llegar puntuales al inicio de la jornada laboral
        Mientras no sea fin de la jornada laboral
                Dirigirnos a un ropero
                Si esta cerrado
                        Abrirlo
                Tomar una camisa
                Si está abrochada
                        Desabrocharla
                        Abrir la camisa
                Si está doblada
                        Desdoblarla
                Meter un brazo por una de sus mangas
                Meter el otro brazo por la otra de sus mangas
                Ajustar la camisa al tronco
                Si es una camisa de botones
                        Abotonarla (botón a botón)
                        Ajustarla al cuerpo
                Sino
                        Ajustarla de manera que quede bien puesta
                Emitir el concepto de calidad sobre la camisa
        Fin_Mientras
Fin
```

Llevándolo a su equivalente en Diagramación Rectangular Estructurada obtendríamos el siguiente Diagrama:



Es importante anotar que cuando exista una decisión podremos tener o no acciones o instrucciones por alguno de sus ramales. Con ello quiero decir que es absolutamente correcto tener una decisión que tenga acciones en el caso de que sea Verdadera la condición y no tenga nada en el caso de que sea Falsa dicha condición.

Cabe anotar que también es correcto tener una decisión dentro de otra decisión ó tener un ciclo dentro de otro ciclo o tener una decisión de un ciclo o un ciclo dentro de una decisión puesto que ninguna de las estructuras son excluyentes.

SeudoCódigos

La tercera técnica para representar algoritmos es la mas obvia y seguramente usted no le va a encontrar nada nuevo: es la técnica de los Seudocódigos. Qué es pues un Seudocódigo..? Pues sencillamente es la representación textual de un algoritmo de manera que dicho texto se encuentre enmarcado en algunas normas técnicas que faciliten su posterior transcripción a un lenguaje de Programación.

Por lo dicho anteriormente es evidente que la técnica de Seudocódigo está muy orientada hacia los algoritmos computacionales. Cuando se habla de algunas normas estamos diciendo que existen unos requisitos que, si bien pueden ser violados, facilitan la posterior transcripción del algoritmo a un Lenguaje de programación de ser cumplidos a cabalidad. No debemos perder el faro que todas estas técnicas nos deben facilitar la posterior transcripción de los algoritmos.

Para escribir pues un algoritmo bajo la forma de seudocódigo algunas de las normas son las siguientes:

Primera Norma.- Siempre se le ha de colocar un nombre al algoritmo de manera que sea lo primero que se lea. Es conveniente acostumbrarse a que dicho nombre no supere los ocho caracteres y preferiblemente que sea altamente mnemónico o sea que su nombre haga una referencia aproximada a lo que contiene. Si a un seudocódigo lo llamamos Seudocódigo X es posible que mas adelante no nos sea muy claro su objetivo pero si lo llamamos seudocódigo Liquidar es muy factible que cada que lo veamos nos vamos a acordar que su objetivo era la liquidación de un determinado valor. Pero si lo llamamos Seudocódigo LiqSalNe es muy posible que cada que veamos este nombre nos acordemos que ese seudocódigo es el que nos permite Liquidar el Salario Neto. Resulta ser muy conveniente por todas las razones que usted se imagine que el nombre de los algoritmos expresados en seudocódigos sea lo mas mnemónicos posibles pues no sabemos cuando tengamos que retomarlos y es allí en donde vamos a tener ver la gran importancia del buen nombre de un algoritmo.

Segunda Norma.- Luego de colocado el nombre del seudocódigo debemos a continuación declarar las variables con las cuales vamos a trabajar durante el programa. Todas las variables que vayan a ser utilizadas deben ser declaradas. Declararlas significa escribir el tipo de dato que van a almacenar y el nombre que dichas variables van a llevar. No se olvide que vamos a trabajar con tres tipos standard de datos como son los datos de tipo entero, datos de tipo real y datos de tipo carácter que tienen cada uno unas características y unas restricciones.

Tercera Norma.- Todo el cuerpo del algoritmo deberá ir “encerrado” entre las palabras *Inicio* y *Fin* indicando en donde comienza y en donde termina el seudocódigo.

Cuarta Norma.-

a. Cuando quiera que salga un título en la pantalla todo lo que tiene que hacer es utilizar la orden *Escriba* y a continuación colocar entre comillas dobles lo que quiera que salga en pantalla. Por ejemplo

   Escriba “Esta es una demostración”

   Generará en pantalla el título *Esta es una demostración*

b. Si usted quiere que lo que salga en pantalla sea el contenido de una variable todo lo que tiene que hacer es utilizar la orden *Escriba* y a continuación y sin comillas dobles el nombre de la variable que quiere escribir. Por ejemplo

   N = 5
   Escriba N

   Mostrará en pantalla el valor *5*

c. Si quiere que en pantalla salga un título y a continuación salga el contenido de la variable todo lo que tiene que hacer es colocar el título entre comillas dobles y, luego de haberlas cerrado, colocar el nombre de la variable que usted quiere escribir al lado del título. Por ejemplo

   N = 8
   Escriba “ El valor es “ N

   Generará en pantalla *El valor es 8*

d. Si quiere mostrar en pantalla el contenido de varias variables entonces simplemente a continuación de la orden Escriba y separadas por comas puede escribir los nombres de las variables que usted quiere escribir. Por ejemplo

   N = 8
   M = 4
   Escriba “Los valores son “ N, M

   Escribirá en pantalla *Los valores son 8 4*

Quinta Norma.-

a. Cuando usted vaya a leer un dato para que sea almacenado en una variable determinada utilice la orden Lea. Para no tener que escribir (por ejemplo)

   ***Lea** un dato entero y guárdelo en la variable **N** que también es entera*

   Solo tiene que escribir

   Lea N

   y el computador lo entenderá correctamente.

b. Cuando necesite leer mas de un dato para ser almacenado en diferentes variables todo lo que tiene que hacer es utilizar la orden Lea y escribir las variables separadas por comas. Por ejemplo

   Lea a, b

   Suponiendo que tanto *a* como *b* son variables de tipo entero, esta orden le indicará al computador que lea un dato entero y lo almacene en la variable *a* y luego que lea otro dato entero y lo almacene en la variable *b*.

c. No necesariamente cuando se lean dos o mas variables utilizando una sola orden Lea, éstas deben ser del mismo tipo. Por ejemplo

   Lea var_e, var_r, var_c

   Asumiendo que *var_e* es una variable de tipo entero, *var_r* es una variable de tipo real y *var_c* es una variable de tipo carácter, esta orden le indicará al computador que lea un valor entero y lo almacene en la variable *var_e*, luego que lea un valor real y lo almacene en la variable *var_r* y luego que lea un carácter y lo almacene en la variable *var_c*.

Sexta Norma.- Cuando necesite tomar una decisión deberá utilizar la orden ***Si***, a continuación escribir la *condición* correspondiente y luego las instrucciones que se han de realizar en caso de que la condición sea Verdadera. En caso de que la condición sea Falsa y tenga instrucciones a realizarse cuando así sea entonces deberá existir una alternativa ***Sino***. Al finalizar toda la decisión deberá existir un indicador ***Fin_Si***. La estructura entonces será la siguiente

```
Si  Condición
        .
        .
        Instrucciones a ejecutar si la condición es Verdadera
        .
        .
Sino
        .
        .
        Instrucciones a ejecutar si la condición es Falsa
        .
        .
Fin_Si
```

Las condiciones pueden ser expresadas utilizando los siguientes ***Operadores Relacionales*** que son los símbolos que nos van a permitir obtener una respuesta Verdadera o Falsa:

>	Mayor que
<	Menor que
>=	Mayor o igual a
<=	Menor o igual a

=	Igual a (también llamado igual de comparación)
< >	Diferente de

Es importante anotar que el signo igual (=) utilizado como operador relacional tiene una función diferente que el signo igual (=) utilizado para asignarle un valor a una variable. En el primer caso sería utilizado para preguntar, por ejemplo, si el contenido de la variable ***a*** es igual al valor 5 y en el segundo caso estaríamos asignando a la variable ***a*** el valor 5. Son dos usos realmente diferentes.

Algunos lenguajes diferencian sintácticamente el igual de comparación del igual de asignación precisamente para que el compilador no tenga la opción de realizar interpretaciones ambiguas. En este libro será claro que la utilización del igual de asignación será en instrucciones de asignación y el igual de comparación será utilizado en decisiones.

También podemos conectar expresiones relacionales (tales como a<b) a través de los ***Operadores Booleanos*** que no son mas que unos signos que nos van a permitir expresar dichas relaciones.

Cuando éramos niños y nos decían, al mandarnos a la tienda, *tráigame una gaseosa y un pan de $ 100,oo.* teníamos varias opciones.

a. Si no traíamos ninguna de las dos cosas entonces no habíamos cumplido la orden.

b. Si no traíamos la gaseosa pero sí traíamos el pan de $100,oo tampoco habíamos cumplido la orden
c. Si traíamos la gaseosa pero no traíamos el pan de $100,oo tampoco habíamos cumplido la orden
d. Si traíamos la gaseosa y también traíamos el pan de $100,oo entonces allí sí habíamos cumplido la orden completamente.

Para este ejemplo, y tal como lo hacía nuestra progenitora, asumimos que cumplir la orden es hacer el "mandado" completo. Igualmente vamos a asumir la frase Hemos cumplido la orden como un Verdadero (V) y la frase No hemos cumplido la orden como un Falso (F). De esta forma podríamos organizar la siguiente tabla de verdad:

Condición 1 ***Tráigame una Gaseosa***	**Condición 2** ***Tráigame un pan de $100,oo***	***Cond1*** **Y** ***Cond2***	**Explicación textual**
F	F	F	No cumplimos completamente la orden
F	V	F	No cumplimos completamente la orden
V	F	F	No cumplimos completamente la orden
V	V	V	No cumplimos

			completamente la orden

Puede notar usted que solamente habremos cumplido la orden completamente si hemos cumplido cada una de las ordenes o de lo que aquí hemos llamado *Condición 1* y *Condición 2*. Igualmente puede usted notar que en el título de la tercera columna hay una **Y** un poco mas grande que el resto del texto. Precisamente esa es el primer operador booleano que vamos a conocer. Es el operador **Y** que en la mayoría de los libros se conoce como el operador **AND** y que precisamente aquí lo vamos a llamar **AND**.

Su tabla de verdad es la que está expuesta en ese ejemplo por lo cual podemos concluir que cuando utilizamos un operador **AND** solamente genera *Verdadero* si ambas condiciones se cumplen (no olvide que se habla de ambas condiciones porque el operador AND puede "conectar" solamente dos condiciones) y en cualquier otro caso genera *Falso*.

Cuando nos decían *Tráigame una Coca-Cola Litro ó una Naranja Postobón Litro* nos estaban dando las siguiente opciones:

a. Si no traemos ninguna de las dos gaseosas entonces no hemos cumplido la orden
b. Si no traemos la Coca-Cola Litro y traemos la Naranja Postobón Litro entonces hemos cumplido la orden
c. Si traemos la Coca-Cola Litro y no traemos la Naranja Postobón Litro entonces hemos cumplido la orden
d. Si traemos ambas gaseosas hemos cumplido sobradamente la orden

Podemos pues con este razonamiento organizar la siguiente tabla de verdad (asumiendo que cumplir la orden se representará con Verdadero y no cumplirla se representará con Falso).

Condición 1 ***Tráigame una Coca-Cola Litro***	**Condición 2** ***Tráigame una Naranja Postobón Litro***	***Cond1* O *Cond2***	**Explicación Textual**
F	F	F	No cumplimos la orden
F	V	V	Cumplimos la orden
V	F	V	Cumplimos la orden
V	V	V	Requete cumplimos la orden

Puede notar que en la tercera columna de esta tabla hay una **O** que une a la Condición1 con la Condición2 y que es precisamente el nombre del segundo operador booleano. Técnicamente lo vamos a llamar **OR**. Cuando el operador OR une dos condiciones, toda la expresión es verdadera si, al menos, una de las dos es verdadera. Es obvio pensar que en el caso en que las dos condiciones sean verdadera entonces todas la expresión será mas que Verdadera.

El tercer operador booleano es el operador **NOT**, este operador actúa sobre una sola expresión y lo que hace es que invierte el sentido de la Condición, es decir, cuando el operador NOT va antes de una condición entonces toda la expresión será verdadera si deja de cumplirse la condición.

Veámoslo con un ejemplo:

```
A = 10
Si NOT( A  = 12 )
```

En la primera línea estamos asignando el valor 10 a la variable A y en la segunda línea estamos preguntando que *si A no es igual a 12,* condición que es Verdadera debido a que la variable A es igual a 10. Luego cuando escribimos la siguiente condición

```
Si NOT ( A > B )
```

Es como si hubiéramos escrito

```
Si ( A < = B )
```

O sea, dicho textualmente, cuándo A no es mayor que B..? Pues cuando es Menor o Igual a B.

Con estos tres operadores booleanos podemos construir una gran cantidad de decisiones y permitir que el computador las ejecute correctamente. Cabe anotar que evaluar una decisión y determinar si es Verdadera o Falsa es lo que mas tiempo le toma a un computador, cuando dicha decisión está implementada en un Lenguaje de Programación.

Séptima Norma.- Se utilizará como estructura de ciclo básica la siguiente

Mientras Condición Haga
.....
.....
.....
Cuerpo del ciclo
.....
.....
Fin_Mientras

Algunos libros escriben *Mientras que* en vez de *Mientras* solo pero esas minucias no son de importancia pues obedecen mas a capricho de autores que a alguna razón de fondo frente a la lógica. En el Cuerpo del Ciclo se colocan las ordenes que se van a repetir (o iterar) mientras la condición sea Verdadera. El Fin_Mientras le indicará posteriormente hasta donde llega el bloque de instrucciones u órdenes y determinar a partir de donde se devuelve el control del algoritmo para evaluar la condición. La forma de ejecución de los ciclos se explicará apropiadamente en el capítulo de ciclos así como las otras formas referenciales que existen para expresar ciclos.

Octava Norma.- Cada que usted vaya a utilizar un conjunto de instrucciones deberá indicar claramente en donde comienza ese conjunto de instrucciones y en donde termina utilizando apropiadamente las palabras *Inicio* y *Fin*.

Tal vez usted estará esperando que escribamos los dos ejemplos iniciales en la forma de Seudocódigo. Efectivamente lo voy a hacer a continuación pero espero que usted vea que en algoritmos informales la utilización del seudocódigo no es muy práctico ya que no existe mucha diferencia entre el algoritmo como tal y su respectivo equivalente en Seudocódigo (excepto algunos detalles que son mínimos comparados con las otras técnicas). Verá una gran utilidad cuando estemos escribiendo algoritmos computacionales utilizando esta técnica. Por ahora la versión en Seudocódigo que le podría brindar acerca de los dos algoritmos es la siguiente: Recuerde que el objetivo era realizar un algoritmo que nos permitiera Vigilar por una ventana asomándonos por ella cada media hora por lo tanto la versión de esta algoritmo en seudocódigo es la siguiente

Algoritmo para Vigilar desde una ventana
Inicio
Llegar puntual a la hora de inicio de la jornada laboral
Ubicarnos en nuestro escritorio
Mientras no sea fin del día
Ubicar la ventana por la que nos queremos asomar
Si estamos sentados
Levantarnos del lugar en donde estemos sentados
Orientarnos hacia la ventana
Sino
Orientarnos hacia la ventana
Avanzar hacia la ventana
Llegar hasta tener la ventana muy muy cerquita
Si esta cerrada
Abrirla

```
                Asomarnos por la ventana
                Regresar a nuestro escritorio
                Mientras no haya pasado Media Hora
                        Permanecer en nuestro escritorio
                Fin_Mientras
        Fin_Mientras
Fin
```

Y para el segundo algoritmo en donde el objetivo era Inspeccionar las camisas en un almacén de ropa emitiendo nuestro concepto acerca de su calidad, la solución es la siguiente:

```
Algoritmo para Inspeccionar las camisas en un almacén de ropa
Inicio
        Llegar puntuales al inicio de la jornada laboral
        Mientras no sea fin de la jornada laboral
                Dirigirnos a un ropero
                Si esta cerrado
                        Abrirlo
                Tomar una camisa
                Si está abrochada
                        Desabrocharla
                        Abrir la camisa
                Si está doblada
                        Desdoblarla
                Meter un brazo por una de sus mangas
                Meter el otro brazo por la otra de sus mangas
                Ajustar la camisa al tronco
                Si es una camisa de botones
                        Abotonarla (botón a botón)
                        Ajustarla al cuerpo
                Sino
                        Ajustarla de manera que quede bien puesta
                Emitir el concepto de calidad sobre la camisa
        Fin_Mientras
Fin
```

Qué hay de diferente...? Pues nada, porque en este tipo de algoritmos los seudocódigos (como le dije en un párrafo anterior) son de muy poca utilidad. No se olvide que la técnica de los seudocódigos está diseñada fundamentalmente para ser utilizados en algoritmos computacionales.

Es por eso que en este momento tal vez quede en su momento una nebulosa de dudas acerca de la utilidad de la técnica en sí, pero no hay nada de qué preocuparse pues precisamente la utilización eficiente de esta técnica será unos de los objetivos fundamentales de este libro para que usted a través de ella pueda expresar cualquier algoritmo computacional y obtenga una solución que luego sea fácilmente codificable en cualquier Lenguaje de Programación.

Cuadro Comparativo

Nombre de la Técnica	Ventajas	Desventajas
Diagramas de Flujo	a. Permite visualizar gráfica-mente el camino que sigue la solución a un problema b. Por ser tan simplificado es muy entendible c. No se necesitan muchos conocimientos técnicos para utilizar esta técnica	a. Dado que los flujos (representados con fle-chas) pueden ir de cualquier lugar a cualquier lugar da espacio para que el diagrama llegue a ser casi inentendible b. Deben conocerse bien los símbolos que se van a utilizar c. No todos los símbolos están estandarizados d. Los ciclos deben ser reinterpretados para poder ser diagramados en esta técnica e. No siempre es muy entendible f. Algunas veces la analogía entre el diagrama y la codificación en el Lengua-je de Programación resul-ta ser compleja
Diagramación Rectangular Estructurada	a. Permite tener un marco referencial concreto y definido para la repre-sentación de los algorit-mos b. Solo tiene tres esquemas que le permiten a su vez representar las tres estructuras básicas c. Exige orden en la representación de un algoritmo d. Es muy entendible e. La analogía entre la codificación y el diagrama normalmente es directa y por lo tanto muy sencilla	a. Exige una fundamen-tación técnica que permita representar la solución a cualquier problema a través de las tres estructuras básicas b. No una técnica muy popularizada
SeudoCódigo	a. Permite expresar la solución algorítmica a un problema en nuestro propio lenguaje y casi con nuestras propias reglas b. La codificación se facilita demasiado dado que la transcripción es directa c. Si el programador es ordenado, esta puede llegar a ser la técnica mas entendible	a. Exige mucho orden para ser utilizada eficiente-mente b. Exige el mantenimiento claro de los conceptos de algoritmos como tales c. Las decisiones deben estar encasilladas dentro de los alcances de los operadores lógicos y operadores booleanos

Aún a pesar de que algunos libros y algunos profesionales de la programación aceptan única y exclusivamente la técnica de los seudocódigos, mi concepto personal es que cada una de estas técnicas tiene unas ventajas y unas desventajas que las hacen comparable con las demás. En mi concepto personal considero de suprema importancia darle prioridad de uso a la técnica que facilite la codificación ya que el computador no ejecutará los algoritmos escritos en estas técnicas sino escritos en términos de un Lenguaje de Programación y puedo garantizar que la técnica que mas facilita la transcripción es el SeudoCódigo sin desconocer las ventajas de cada una de las otras técnicas.

Ejercicios

Utilizando las técnicas explicadas REPRESENTAR los siguientes algoritmos:

1. Desarrollar un algoritmo que permita adquirir una revista.
2. Desarrollar un algoritmo que permita entrar a una casa que está con llave.
3. Desarrollar un algoritmo que permita dar un beso.
4. Desarrollar un algoritmo que permita empacar un regalo.
5. Desarrollar un algoritmo que permita encender un vehículo.
6. Desarrollar un algoritmo que permita fritar un huevo.
7. Desarrollar un algoritmo que permita mirar por un telescopio.
8. Desarrollar un algoritmo que permita botar la basura.
9. Desarrollar un algoritmo que permita tomar un baño.
10. Desarrollar un algoritmo que permita estudiar para un examen.
11. Desarrollar un algoritmo que permita tocar determinada canción con un instrumento musical.
12. Desarrollar un algoritmo que permita viajar en avión.
13. Desarrollar un algoritmo que permita encender un bombillo.
14. Desarrollar un algoritmo que permita encender una vela.
15. Desarrollar un algoritmo que permita apagar una vela.
16. Desarrollar un algoritmo que permita apagar un bombillo.
17. Desarrollar un algoritmo que permita parquear un vehículo.
18. Desarrollar un algoritmo que permita almorzar.
19. Desarrollar un algoritmo que permita ir de la casa al trabajo.
20. Desarrollar un algoritmo que permita colocarse una camisa.
21. Desarrollar un algoritmo que permita quitarse la camisa.

22. Desarrollar un algoritmo que permita escuchar un determinado disco.
23. Desarrollar un algoritmo que permita abrir una ventana.
24. Desarrollar un algoritmo que permita ir a la tienda a comprar algo.
25. Desarrollar un algoritmo que permita tomar una fotografía.
26. Desarrollar un algoritmo que permita hacer deporte.
27. Desarrollar un algoritmo que permita cortarse el cabello.
28. Desarrollar un algoritmo que permita hacer un avión con una hoja de papel.
29. Desarrollar un algoritmo que permita manejar una bicicleta.
30. Desarrollar un algoritmo que permita manejar una motocicleta.
31. Desarrollar un algoritmo que permita manejar un monociclo.
32. Desarrollar un algoritmo que permita maquillarse.
33. Desarrollar un algoritmo que permita hacer un pastel.
34. Desarrollar un algoritmo que permita hacer un almuerzo.
35. Desarrollar un algoritmo que permita adquirir un pantalón.
36. Desarrollar un algoritmo que permita hacer un mercado pequeño.
37. Desarrollar un algoritmo que permita leer el periódico.
38. Desarrollar un algoritmo que permita saludar a un amigo.
39. Desarrollar un algoritmo que permita arrullar a un bebé hasta que se duerma.
40. Desarrollar un algoritmo que permita hacer un gol en fútbol.
41. Desarrollar un algoritmo que permita jugar ping-pong.
42. Desarrollar un algoritmo que permita nadar.
43. Desarrollar un algoritmo que permita tirarse desde un avión con un paracaídas.
44. Desarrollar un algoritmo que permita tirarse desde un avión sin un paracaídas.
45. Desarrollar un algoritmo que permita descifrar un jeroglífico.
46. Desarrollar un algoritmo que permita amarrase un zapato.
47. Desarrollar un algoritmo que permita quitarse los zapatos.
48. Desarrollar un algoritmo que permita silbar.
49. Desarrollar un algoritmo que permita elevar una cometa.

50. Desarrollar un algoritmo que permita desarrollar algoritmos.