Tipos de Estructuras

Las Expresiones y Operaciones que se usan en un algoritmo se estructuran de diversas formas. Este capítulo muestra cuales son esas diferentes formas y para que se usan.

3.1 Tipos de Estructuras

El concepto *Programación Estructurada* no siempre ha estado presente en la programación de computadoras, es a lo largo de las aproximadamente cinco décadas de existencia de la computación que se han desarrollado y encontrado paulatinamente nuevas y mejores formas de programar.

El concepto *Programación Estructurada* involucra una forma de programar basada en estructuras que se van sucediendo unas después de otras o unas dentro de otras. La ventaja de resolver los problemas de esta forma es que estarán muy identificados los diferentes grupos de acciones que deben realizarse para resolver el problema, serán fáciles de construir, fáciles de leer y por lo tanto fáciles de modificar.

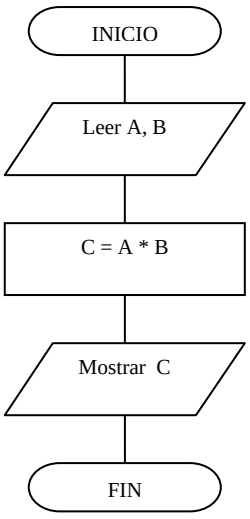
Las principales estructuras que se pueden plantear son:

- Estructura lineal
- Estructura de bifurcación
- Estructura de repetición

No necesariamente vamos a encontrar solo una de estas estructuras en un algoritmo, en realidad se van sucediendo unas después de otras e incluso unas dentro de otras. Para efectos del estudio de las mismas, primero vamos a verlas en forma independiente para luego irlas combinando poco a poco.

3.2 Estructura Lineal

En esta forma las acciones ocurren una después de la otra. El flujo es una sola línea que va encontrando en su camino diversas acciones. No hay derivaciones y es fácilmente identificable el inicio y el final de la estructura. En este tipo de estructura no vemos expresiones booleanas.



No es muy común encontrar un algoritmo que solo este constituido por una de estas estructuras, en todo caso, de existir un algoritmo así, es sumamente simple y su trabajo es muy elemental. Más bien suelen encontrarse estas estructuras como una parte de un algoritmo que es mucho más grande.

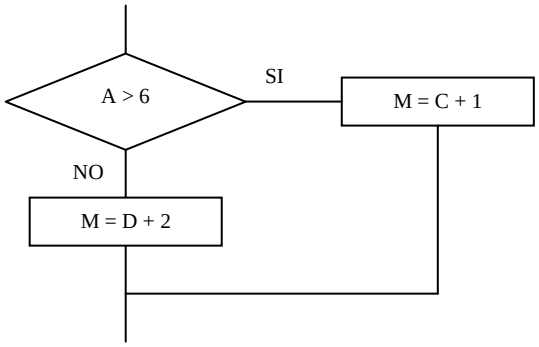
En el diagrama al costado vemos los terminales, es decir el inicio y fin de un algoritmo. La segunda acción es una operación de entrada, donde se leen valores para las variables A y B. Luego vemos una expresión aritmética que coloca en la variable C el resultado de la multiplicación de los valores de A por B. Luego la operación de salida que muestra Finalmente el terminal que da por concluido el algoritmo. el valor de la variable C.

```
Begin
Read A, B
C = A * B
Write C
End
```

Este algoritmo también puede ser expresado en pseudocódigo, como lo muestra el recuadro de la izquierda.

3.3 Estructura de Bifurcación

A partir de esta estructura, los ejemplos mostrados no serán algoritmos completos, sino parte de los mismos, debemos por lo tanto suponer que el flujo llega al ejemplo luego de haber realizado otras acciones que no importan para el caso, así mismo, sale del ejemplo y continúa con otras acciones. Cuando recorremos una carretera y encontramos un lugar donde el camino se parte en dos, estamos frente a una bifurcación, el prefijo *Bi* significa dos. De igual forma ocurre con el flujo, en determinado momento se presenta una condición que hace que el flujo se divida en dos.



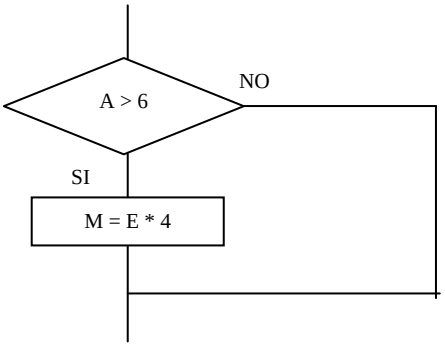
Esta estructura usa como elemento base a una expresión booleana, es decir una condición o pregunta. La condición está preguntando si el valor de A en ese momento es mayor a 6. Solo existen dos posibles respuestas: Si o No. Si la respuesta es Si entonces el flujo continuarán por la derecha, se ejecutará la expresión M = C + 1. En cambio, si la respuesta es No entonces el flujo continuará hacia abajo y se ejecutará la expresión M = D + 2. Debemos tener en cuenta que las estructuras de bifurcación siempre tendrán un final. En el diagrama, el final de la estructura es el lugar donde vuelven a encontrarse las líneas del flujo que se separaron.

```
If A > 6
    Then
        M = C + 1
    Else
        M = D + 2
End
```

En el recuadro de la izquierda se muestra la estructura expresada en pseudocódigo. Tome en cuenta que se están empleando márgenes diferentes para definir que está dentro de que. En un mismo margen están el inicio y el final de la estructura, luego en otro margen el *Then* y el *Else*, finalmente en otro margen los contenidos del *Then* y el *Else*. El *End* equivale al lugar donde las líneas vuelven a juntarse en el diagrama.

En cuanto al diagrama es importante comentar que no existen reglas para determinar hacia donde serán las salidas del Si y el No. La estructura propuesta podría haber tenido la salida del Si hacia la izquierda y la del No a la derecha, o la del Si hacia abajo y la del No hacia la derecha. Cuál sea la forma escogida dependerá del diagrama completo, alguna de las formas proporcionará una mayor comodidad al momento de diseñar todo el algoritmo.

Las estructuras de bifurcación pueden plantearse también de forma que solo tienen acciones en la salida del Si y nada en la salida del No, tal como se muestra en la estructura de la derecha.



En este caso se pregunta si el valor de A es mayor a 6. Si la respuesta es Si, entonces se ejecuta la expresión aritmética M = E * 4. Si la respuesta es No entonces no se ejecuta nada.

```
If A > 6
    Then
        M = E * 4
End
```

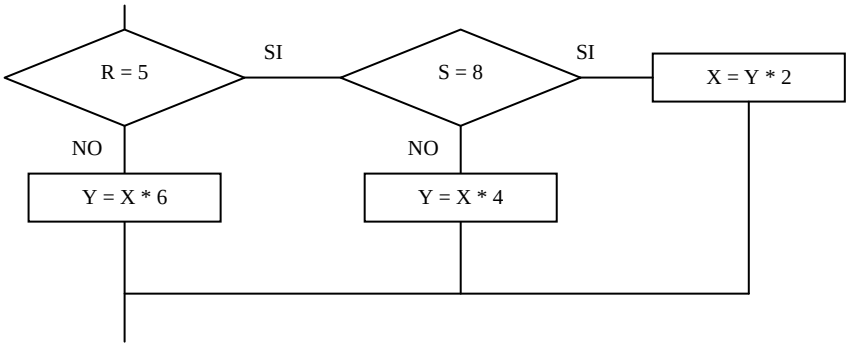
Esta estructura se expresa en pseudocódigo tal se muestra en el recuadro de la izquierda. En este caso no se escribe el *Else* pues no contendrá nada. Es muy importante tener en cuenta que no debe

plantearse el caso opuesto, es decir una estructura que tenga acciones en el No y el Si se encuentre vacía. No es posible escribir una bifurcación con *Else* pero sin *Then*.

3.4 Estructura de Bifurcación anidada

El término *anidado* es posible encontrarlo en muchos aspectos relacionados con la computación. Entendemos *anidado* como un elemento que está dentro de otro similar o de la misma índole.

En los casos anteriores las salidas del *Si* y el *No* han planteado solamente expresiones aritméticas en su contenido, en cambio ahora vamos a plantear otra expresión booleana dentro de una de las salidas.

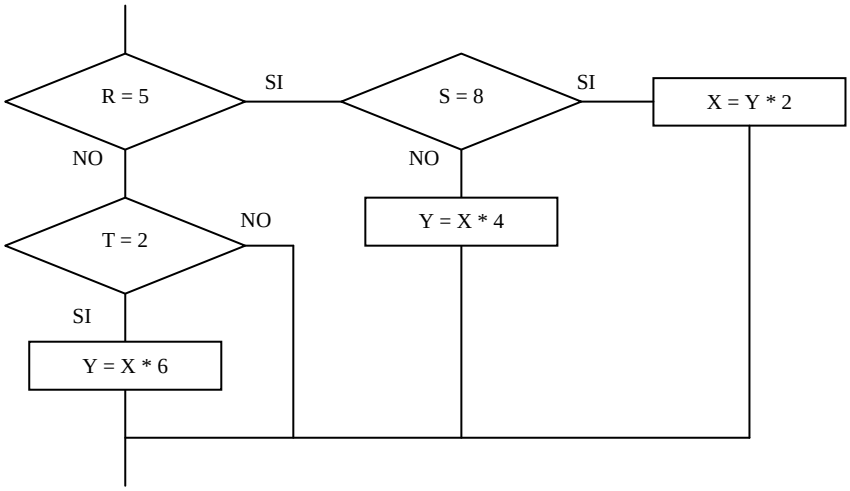



```
If R = 5
    Then
        If S = 8
            Then
                X = Y * 2
            Else
                Y = X * 4
        End if
    Else
        Y = X * 6
End if
```

Si el valor de R es igual a 5 entonces se preguntará inmediatamente si el valor de S es igual a 8, esta expresión es parte del *Si* de la anterior, pero tiene a su vez sus propios *Si* y *No*. La forma de expresar esta estructura en pseudocódigo es la que se muestra a la izquierda.

En este caso debe tenerse en cuenta nuevamente el uso de los diferentes márgenes para escribir claramente los textos. Cuando anidamos estructuras entonces debe tenerse en cuenta como se cierran. Se cierra primero la última que se abrió. Esto se ve claramente en el pseudocódigo, la pregunta S = 8 ha sido cerrada antes que la pregunta R = 5. Esta situación no es tan fácil de expresar en el diagrama. Pueden haber anidados dentro de anidados, por lo tanto, si se abren las estructuras 1, 2, 3, 4, entonces deben cerrarse en el orden 4, 3, 2, 1.

En la estructura anterior encontramos un anidado en la salida del *Si*, pero también podrían haber anidados en ambas salidas, como lo muestra el diagrama de la derecha. En este caso vemos tres estructuras de bifurcación, dos de ellas anidadas en otra. La forma de expresar está estructura en pseudocódigo es la que se muestra en el texto siguiente.



```
If R = 5
    Then
        If S = 8
            Then
                X = Y * 2
            Else
                Y = X * 4
        End if
    Else
        If  T = 2
            Then
                Y = X * 6
        End if
End if
```

Nuevamente vea como se han usado los márgenes diferenciados para escribir más ordenado el texto. La primera estructura abierta es R = 5, luego se abre la estructura S = 8, entonces debe cerrarse primero S = 8. Antes de cerrarse R = 5 se encuentra otro anidado en el *Else*, es decir T =2, por lo tanto primero se cierra T = 2 y finalmente se cierra R = 5.

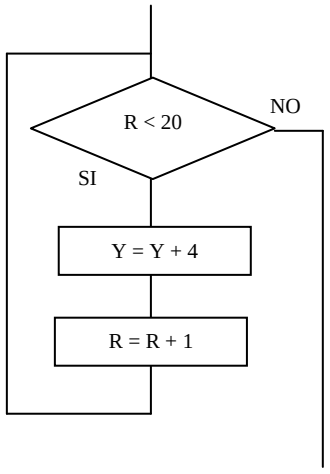
La estructura S = 6 tiene acciones en el *Si* y el *No*, por lo tanto tiene *Then* y *Else*, en cambio, la estructura T = 2 solo tiene acciones en el *Si*, por lo tanto solo tiene *Then*, pero no tiene *Else*. La estructura principal R = 5 tiene acciones tanto en el Si como en el No.

3.5 Estructura de Repetición

Las condiciones o expresiones booleanas no solo se usan para crear estructuras de bifurcación, también se usan para crear estructuras de repetición, es decir situaciones que deben repetirse *n* cantidad de veces.

En principio el trabajo es el mismo, es decir se evalúa la pregunta y tiene dos posibles respuestas, solo que en este caso, una de ellas produce un retorno de forma que una o varias acciones pueden volver a ejecutar nuevamente.

Las repeticiones se pueden plantear de dos formas. En el diagrama de la derecha se ve una repetición de tipo DO WHILE, que quiere decir *hacer mientras suceda algo.* También entendido como *hacer mientras la respuesta a la pregunta sea Si.*



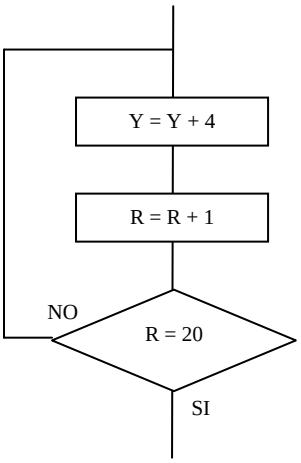
Cuando el flujo llega a la estructura, la variable R tiene algún valor. Supongamos que el valor es 1, entonces se preguntará si el valor de R es menor a 20, la respuesta será Si y por lo tanto el flujo continuará hacia abajo, para ejecutar las dos expresiones aritméticas. Sin embargo note que una de las expresiones está incrementando el valor de M, de forma que la segunda vez que se hace la pregunta el valor de M será 2 y luego 3 y 4, etc.

```
Do while R < 20
     Y = Y + 4
     R = R + 1
Loop
```

Llegará el momento en que el valor de M será 20 y por lo tanto la respuesta será No. En ese momento el flujo continuará por la derecha y saldrá de la estructura de repetición.

La forma de escribir está estructura es la que se muestra en el recuadro de la izquierda. Note que no se usa la palabra *If*, se usa en cambio la palabra *Do*, por lo tanto no existe en esta estructura ni el *Then* ni el *Else*. En el diagrama se expresan los casos del *If* y del *Do* con un rombo, pareciendo que no es posible diferenciarlos, sin embargo, note que en el caso de la bifurcación los flujos del *Si* y el *No* vuelven a juntarse después de terminada la estructura, en cambio en el caso de la repetición no vuelven a juntarse. Esta situación es la que hace la diferencia en el diagrama.

La otra forma de plantear una repetición es la que se muestra en el diagrama de la derecha. Se trata del DO UNTIL, que quiere decir *hacer hasta que suceda algo*, también entendido como *hacer hasta que la respuesta a la pregunta sea Si.*



Cuando el flujo llega a la estructura, la variable R tiene un valor, supongamos que es 0, se ejecutan las dos acciones y luego se pregunta si el valor de R es igual a 20, la respuesta será No, por lo tanto se volverán a ejecutar las dos acciones. Vea que una de las expresiones aritméticas incrementa el valor de M, por lo tanto la segunda vez que se haga la pregunta será 2 y así sucesivamente. Llegará el momento en que el valor de M será 20, momento en el cual la respuesta a la pregunta será Si, y entonces el flujo saldrá de la estructura y continuará con otras acciones. La forma de representar esta estructura en pseudocódigo es la que se muestra a la izquierda.

```
Do until R = 20
    Y = Y + 4
    R = R + 1
Loop
```

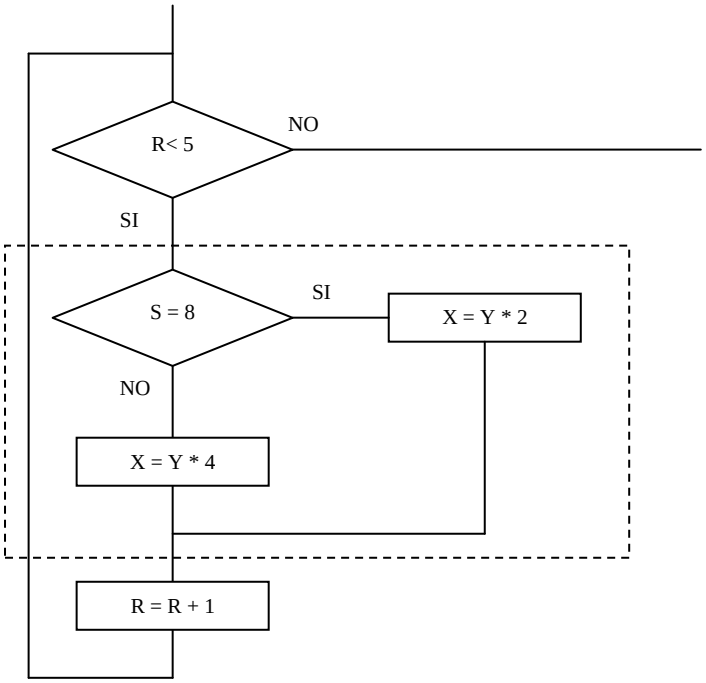
Nuevamente se puede notar la ausencia del *If*, *Then* y *Else*. También se puede notar que los flujos del *Si* y el *No*, no se vuelven a juntar, lo cual diferencia claramente esta estructura, (en el diagrama), de una bifurcación.

Es importante recalcar algunas diferencias entre estos dos tipos de repeticiones:

- El Do WHILE repite por la salida del Si, en cambio el DO UNTIL repite por la salida del No.
- El DO WHILE tiene siempre la pregunta al incio de la repetición, en cambio el DO UNTIL tiene la pregunta al final de la repetición
- El DO WHILE podría no entrar nunca a la repetición ya que la primera vez que se pregunta podría ser negativa la respuesta, en cambio el DO UNTIL siempre entrará por lo menos una vez ya que la pregunta se hace al final.

3.6 Estructuras de bifurcación anidadas en estructuras de repetición

Habiendo conocido como es el funcionamiento de una bifurcación y de una repetición podemos deducir que una puede estar dentro de la otra. En el diagrama de la derecha podemos ver dicho caso. La bifurcación esta marcada dentro de un recuadro, dicho recuadro no es parte del diagrama, se ha colocado solo para dar más claridad al estudiante.



Se puede notar como en la bifurcación los flujos vuelven a encontrarse, en cambio en la repetición no lo hacen.

Se puede notar también que la expresión R = R + 1 está dentro de la

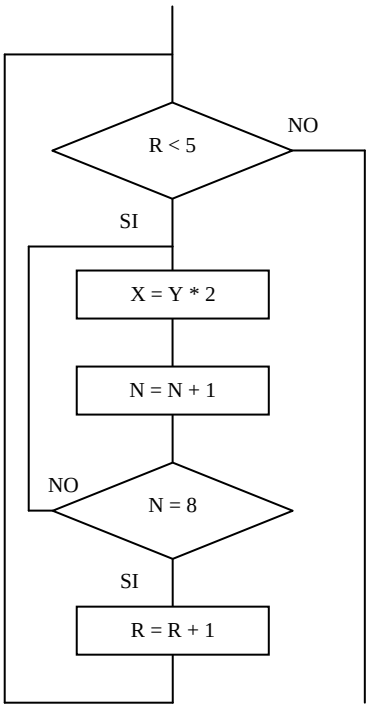
repetición pero está fuera de la bifurcación.

```
Do while R < 5
    If S = 8
        Then
            X = Y * 2
        Else
            X = Y * 4
    End if
    R = R + 1
Loop
```

La forma de escribir esta estructura en pseudocódigo en la que se muestra a la izquierda. Nuevamente se puede notar que primero se cierra la bifurcación pues fue la última en abrirse. La última en cerrarse es la repetición pues se abrió primero.

3.7 Estructuras de repetición anidadas en otras estructuras de repetición

También es posible anidar estructuras de repetición unas dentro de otras. En el caso del diagrama de la derecha tenemos una estructura DO UNTIL, anidada dentro de una estructura DO WHILE.



Se puede notar que en ninguna de los casos de las expresiones booleanas (rombos), los flujos vuelven a encontrarse, está claro entonces que se trata en ambos casos de repeticiones.

```
Do while R < 5
    Do until N = 8
        x = Y * 2
        N = N + 1
    Loop
    R = R + 1
Loop
```

El recuadro de la izquierda nos muestra el pseudocódigo correspondiente a este algoritmo.

Su puede notar como la expresión R = R + 1 está afuera del Do until, pero dentro del Do while.

3.8 Seguimiento y comprensión del funcionamiento de un algoritmo

En los acápites anteriores se han visto ejemplos en los cuales no ha sido necesario analizar que está pasando con los valores de las variables, la atención se ha enfocado en comprender y reconocer las estructuras. Sin embargo, es importante ser capaz de entender también lo que está pasando con las estructuras, es requisito indispensable para poder posteriormente armar soluciones completas a problemas concretos.
En los siguientes capítulos se expondrán algunos casos a través de los cuales podremos ir adquiriendo destreza en este aspecto.

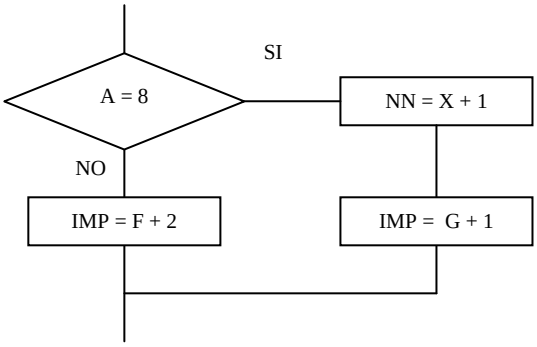
Ejercicios para el alumno

Los siguientes ejercicios tienen como objetivo que el estudiante llegue a tener destreza en reconocer las diferentes estructuras dentro de un algoritmo y a dominar la equivalencia entre un flujograma y un pseudocódigo.

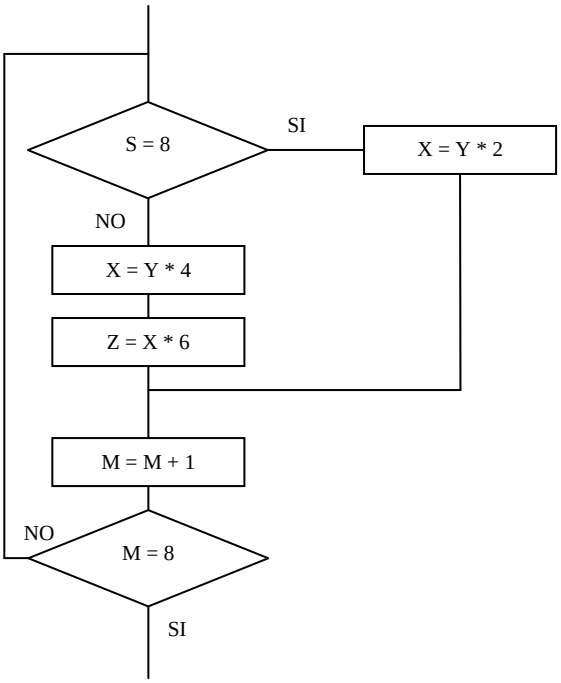
Algunos de los ejercicios están desarrollados, otros se dejan sin solución para que el estudiante los ejecute.

1.- Escribir en Pseudocódigo el diagrama de la derecha.

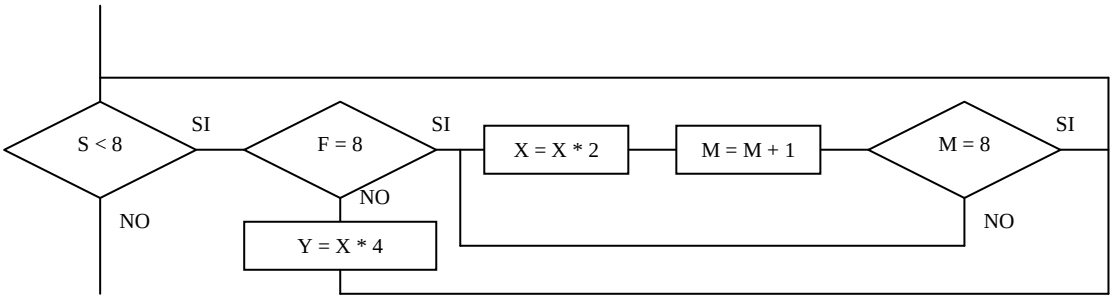
2.- Cambiar el flujograma para que la salida del Si sea hacia abajo y la salida del No hacia la derecha.



3.- Escriba en Pseudocódigo el diagrama de la derecha. En este diagrama hay una repetición y una bifuracción- Para lograr escribir el pesudocódigo debe identifícarlas y determinar cuál está adentro y cual está afuera.



4.- Escriba en pseudocódigo el diagrama de abajo. Hay dos repeticiones y una bifurcación.



:

RESPUESTAS A ALGUNOS DE LOS EJERCICIOS

1.-

```
If A = 8
    Then
        NN = X + 1
        IMP = G + 1
    Else
        IMP = F + 2
End if
```

3.- La expresión booleana que se encuentra al inicio es una bifurcación, los dos flujos vuelven a encontrarse. La expresión booleana que se encuentra al final es una repetición Do Until. La primera está anidada dentro de la segunda.

```
Do until M = 8
    If S = 8
        Then
            X = Y + 2
        Else
            X = Y * 4
            Z = X * 6
    End if
    M = M + 1
Loop
```