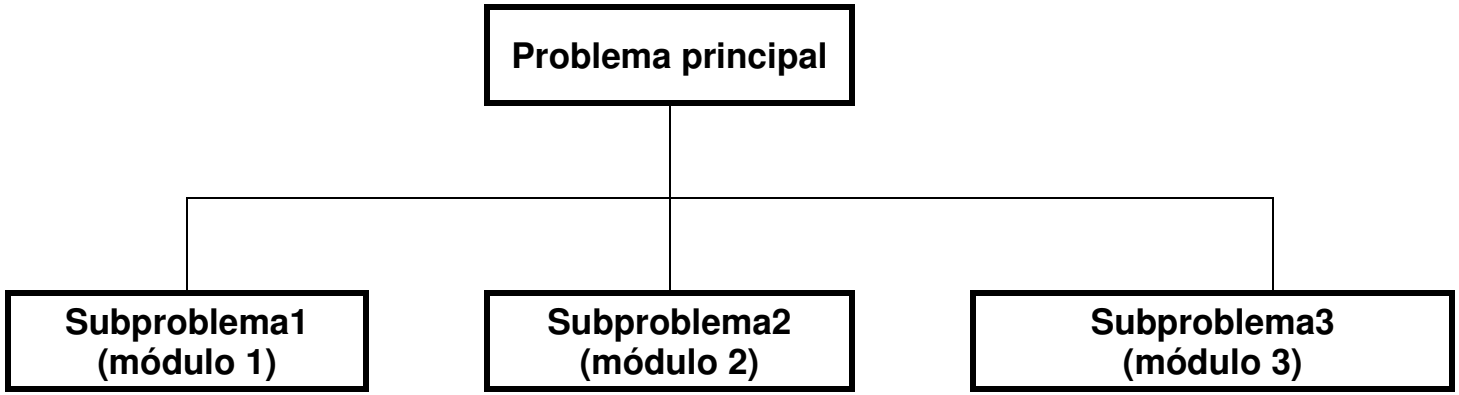
Programación MODULAR: Subalgoritmos - funciones y procedimientos

Uno de los métodos fundamentales para resolver un problema es dividirlo en problemas más pequeños, llamados **subproblemas**. Estos problemas pueden a su vez dividirse repetidamente en problemas más pequeños hasta que los problemas sean de fácil solución. *Divide y vencerás* … Cada subproblema es deseable que sea independiente de los demás y se denomina **módulo**. El problema original se resuelve con un **programa principal** (llamado también driver o main), y los subproblemas (módulos) mediante **subprogramas: *procedimientos*** y ***funciones***.



La resolución de un problema comienza con una descomposición modular y luego nuevas descomposiciones de cada módulo en un proceso denominado refinamiento sucesivo.

Diseño MODULAR: Los módulos

Los subproblemas o módulos se diseñan con **subprogramas**, que a su vez se clasifican en **procedimientos** y **funciones.**

Los procedimientos y funciones son unidades de programas diseñados para ejecutar un a tarea específica. Por ejemplo los procedimientos predefinidos **LEER** y **ESCRIBIR** están diseñados para realizar operaciones de entrada y salida de datos de un programa.

El proceso de descomposición de un problema en módulos se denomina modularización. Los procedimientos y funciones asisten a la programación modular.

- Las funciones, normalmente, devuelven un sólo valor a la unidad de programa (programa que invoca a la función) que las referencia. Los procedimientos pueden devolver cero, uno o varios valores. En el caso de no devolver ningún valor, realiza alguna tarea tal como alguna operación de entrada y/o salida.
- A un procedimiento no se le puede asignar valor, y por consiguiente ningún tipo está asociado con el nombre del procedimiento.
- Una función se referencia utilizando su nombre en una expresión, mientras que un procedimiento se referencia por su llamada o invocación al mismo.

FUNCIONES

Un subalgoritmo función es un subalgoritmo que recibiendo o no datos devuelve un único resultado.

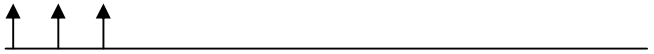
Tienen su origen ligado al concepto matemático de función de una o más variables. Ejemplo de este tipo de subalgoritmos son las llamadas funciones internas (sin, cos, abs). Las cuales pueden usarse en expresiones algorítmicas como si se tratara de variables.

Otros ejemplos de funciones matemáticas:

f(x) = x^2 + 2x – 1

g(x , y) = x^3 + 5y

h(x , y , z) = 3x + 2y - z



parámetros formales

donde **x,y,z**: son los **parámetros formales** o ficticios, es decir permiten expresar la ley o "forma" de la función.

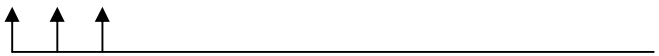
Las funciones pueden tener uno o más parámetros formales (datos) pero siempre devuelven un único resultado.

Las funciones son evaluadas utilizando **parámetros actuales** o reales, es decir los valores con los que se quiere evaluar la función:

f(3)

g(-1 , 5)

h(2 , 0 , 7)



parámetros actuales

FUNCIONES

Una función es un objeto del ambiente, con nombre, tipo y valor único. El tipo se asocia al valor que retorna la función cuando es evaluada para un conjunto dado de valores de sus argumentos

```
Funcion nombre (lista de parámetros formales): Tipo de resultado
    Declaración de variables
Inicio
    Acciones
    Devolver (constante, variable o expresión)
Fin función
```

Lista de parámetros formales: contiene las variables que pasan alguna información necesaria para que la función ejecute el conjunto de acciones.

Tipo de resultado: señala el tipo de dato que devuelve la función.

Declaración de variables: en este lugar se deben declarar los parámetros formales y también aquellas variables que se usarán en la función.

Cuerpo de la función: lo constituye el conjunto de acciones a realizar por la función.

Retornar el resultado: el único resultado que devuelve la función puede ser un valor constante, o una variable o una expresión válida, la cual debe colocarse entre paréntesis al lado de la acción Devolver. Cuando se ejecuta esta acción se devuelve el control del programa al lugar donde se ha llamado a la función.



Ejemplos:

- definición de la función $f(x) = x^2 + 2x - 1$

```
funcion f (x: real): real
inicio
    Devolver ( x * x + 2 * x – 1 )
fin funcion
```

- definición de la función $y = x^n$ (n entero)

```
funcion potencia (x: real, n: entero): real
variables
    entero  i
    real  y
inicio
    y ← 1
    repetir para i ← 1 , abs (n)
        y ← y * x
    fin para
    si  n < 0  entonces
            y ← 1 / y
    fin si
    Devolver ( y )
fin funcion
```

Ejemplo: Escribir un algoritmo que utilice la función del ejemplo anterior para calcular la potencia n de x. Los valores de n y x deberán ser ingresados por teclado, siendo n un entero y x un valor real.

Caso 1

```
algortimo calculo_potencia
variables
    entero : n
    real : w
inicio
    escribir (‘Evalúa la función y = w ^ n ‘)
    escribir (‘Ingrese valor de n’)
    leer (n)
    escribir (‘Ingrese valor de w’)
    leer(w)
    escribir (‘El valor es :’, potencia (w,n) )
fin
```

Caso 2

```
algortimo calculo_potencia
variables
    entero : n
    real : w , z
inicio
    escribir (‘Evalúa la función y = w ^ n ‘)
    escribir (‘Ingrese valor de n’)
    leer (n)
    escribir (‘Ingrese valor de w’)
    leer(w)
    z ← potencia (w,n)
    escribir (‘El valor es :’, z )
fin
```

En el ejemplo 2 se muestran dos formas posibles de utilización de la función potencia: en el caso 1 la función potencia es empleada directamente dentro de la acción escribir, mientras que en el caso 2 el valor devuelto por la función potencia es asignado en la variable real z y es el valor almacenado en z el que luego se muestra.

PROCEDIMIENTOS o SUBRUTINAS

Un procedimiento o subrutina es un subalgoritmo que recibiendo o no datos permite devolver varios resultados, un resultado o ninguno.

Un procedimiento está compuesto por un grupo de sentencias a las que asigna un nombre (identificador o simplemente nombre del procedimiento) y constituye una unidad de programa. La tarea asignada al procedimiento se ejecutará siempre que se encuentre el identificador (nombre del procedimiento) en el conjunto de sentencias que definen el programa.

Cómo trabajar con procedimientos?

- **Declaración de un procedimiento**
- **Llamada a un procedimiento**
- **Dónde escribir un procedimiento?**
- **Transferencia de información a/desde procedimientos: parámetros**
- **Ventajas de utilizar un procedimiento**

Declaración de un procedimiento

La declaración de un procedimiento no indica a la computadora que ejecute las instrucciones dadas, sino que indica a la computadora cuáles son estas instrucciones y dónde están localizadas cuando sea necesario.

Declaración

- **Formato 1**

```
Subrutina nombre()
    Declaración de variables
Inicio
    Acciones
Fin subrutina
```

- **Formato 2**

```
Subrutina nombre (lista de parámetros formales)
    Declaración de variables
Inicio
    Acciones
Fin subrutina
```

Nombre: identificador válido

Lista de parámetros formales: parámetros formales del procedimiento; sirven para pasar información al procedimiento y/o devolver información del procedimiento a la unidad de programa que le invoca.

Están separados por comas, y precedidos por las letras E (entrada), S (Salida) o E/S (Entrada/Salida)



Llamada al procedimiento

Los procedimientos se llaman dentro de un programa o de otro procedimiento directamente por su nombre, de acuerdo a los formatos 1 o 2

- **Formato 1**

```
nombre
```

- **Formato 2**

```
nombre (lista de parámetros formales)
```

La sentencia nombre indica la ejecución del procedimiento cuyo identificador coincide con nombre. Después que ha terminado la ejecución, se ejecuta la sentencia que sigue a la llamada al procedimiento.

En resumen, un procedimiento, al igual que un programa, consta de tres partes:

- Una cabecera del procedimiento que proporciona el nombre del mismo y, caso de existir, una lista de parámetros formales.
- Una sección de declaración que puede contener constantes variables, etc.
- Una sección ejecutable: cuerpo de acciones del procedimiento.



Ejemplos:

- Muestra un texto como encabezado

```
Subrutina Encabezado( )
Inicio
        Escribir ('*******************************')
        Escribir (' INFORMATICA ')
        Escribir ('*******************************')
Fin subrutina
```

- Calcula el área de un círculo

```
Subrutina Superficie(E: r: real, S: A: real)
Inicio
    A ← 3.1415192 * r ^ 2
Fin subrutina
```

Este último procedimiento recibe como información el radio del círculo y devuelve el área del mismo.

r es una variable de entrada: tiene un valor definido previamente (parámetro actual)
A es una variable de salida: no tiene un valor asignado previamente y su valor es definido en por el conjunto de acciones del procedimiento.



Dónde escribir el procedimiento?

La posición adecuada de dónde escribir el procedimiento depende del lenguaje de codificación elegido. En seudocódigo, será indistinto el orden en que se escriben el algoritmo y los subalgoritmos. En este ejemplo, adoptamos arbitrariamente escribirlo luego del programa principal.

```
Programa Círculo
Variables
    Real : radio, area
Inicio
    Escribir('ingrese radio del círculo')
    Leer(radio)

    Superficie(radio, area)

   Escribir('el área del círculo es',area)
Fin

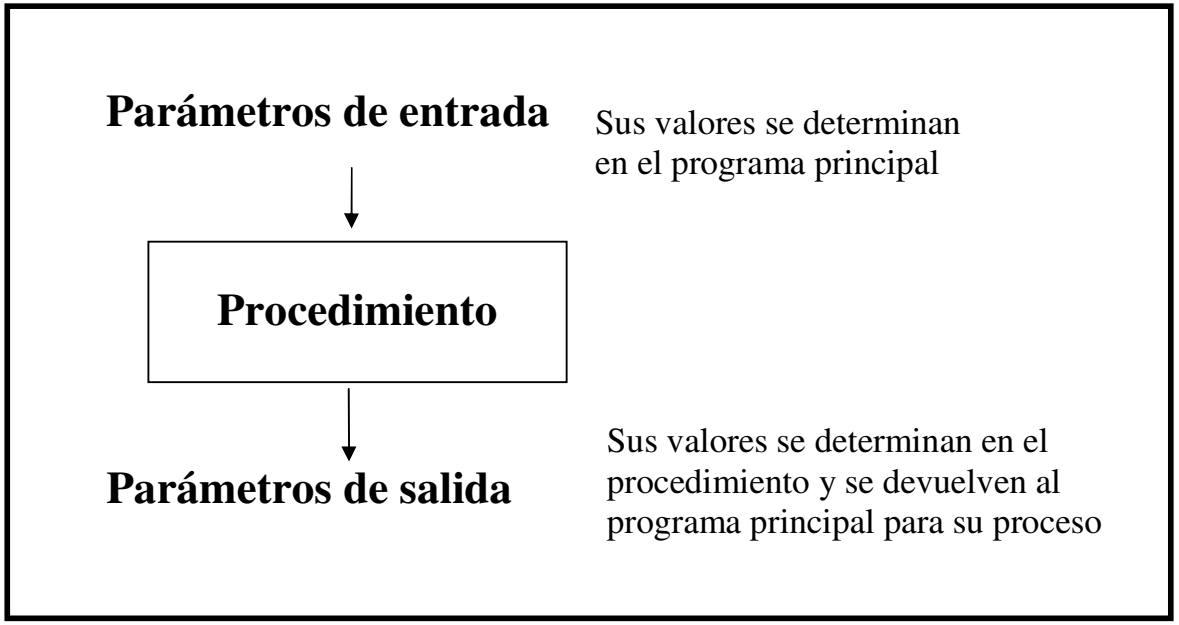
Subrutina Superficie(E: r: real , S: A: real)
Inicio
    A ← 3.1415192 * r ^ 2
Fin subrutina
```

Transferencia de información a/desde procedimientos: parámetros

Un parámetro es un método para pasar información – valores a variables – del programa principal a un procedimiento.

Procedimientos sin parámetros: no existe comunicación entre el programa principal y los procedimientos o entre dos procedimientos.

Procedimientos con parámetros: existe comunicación entre el programa principal y los procedimientos o entre dos procedimientos.



Parámetros actuales y parámetros formales

Las acciones que contienen el llamado al procedimiento constan de dos partes: un identificador (nombre del procedimiento) y una lista de **parámetros actuales**

Nombre (par1, par2, par3, ….)

Los parámetros actuales par1, par2, etc. Contienen los valores que se transferirán al procedimiento.

En la declaración de un procedimiento, cuando se incluyen los parámetros, éstos se denominan **parámetros formales** parf1,parf2, parf3, etc. Ellos sirven para contener los valores de los parámetros actuales cuando se invoca al procedimiento.

Procedimiento Nombre (parf1, parf2, parf3, …)

```
Programa Círculo
Variables
    Real : radio, area
Inicio
    Escribir('ingrese radio del círculo')
    Leer(radio)

    Superficie(radio, area)

   Escribir('el área del círculo es',area)
Fin

Subrutina Superficie(E: r: real , S: A: real )
Inicio
    A ← 3.1415192 * r ^ 2
Fin
```

Los parámetros actuales y formales deben coincidir en números, tipo y orden. Es decir debe existir correspondencia entre los parámetros actuales y formales

Parámetros valor y referencia (variable)

Paso por VALOR: Los valores iniciales se proporcionan copiando los valores correspondientes en la lista de parámetros actuales.

Paso por REFERENCIA: Se produce el paso de la dirección del parámetro actual. En realidad se pasa la posición de memoria. Es decir que una variable pasada por referencia puede ser modificada dentro del subprograma y producir un efecto en el programa de llamada. Utilizaremos la palabra reservada var para indicar este tipo de paso.

```
Algoritmo pasaje
variables
    entero : A,B
Inicio
    A ← 0
    B ← 0
    Escribir (A,B)
    Cambia(A,B)
    Escribir (A,B)
fin

Subrutina Cambia (E: x: real, S:  y: real)
inicio
     Escribir (x,y)
      x ← 1
      y ← 1
     Escribir (x,y)
fin subrutina
```

¿Qué pasa en memoria?

1) Antes de invocar al procedimiento:

A	B		
0	0		

2) Al comenzar el procedimiento:

A	B=Y	X	
0	0	0	

3) Al finalizar el procedimiento:

A	B=Y	X	
0	1	1	

4) Al volver al algoritmo principal:

A	B		
0	1		

Parámetros valor y referencia (variable)

Algoritmo EJEMPLO **variables** **entero : A,B,C** **Inicio** A ← 3 B ← 5 C ←17 SUMAR (A, A, A+B, C) **Escribir** (‘el valor en C es :’, C) **fin**	• **Por valor** **Subrutina** SUMAR **(E:** x, y, z, v : entero**)** **inicio** x ← x +1 v ← y + z **fin subrutina** • **Por referencia** **Subrutina** SUMAR **(E/S:** x, y : entero, **E:** z: entero, **E/S:** v: entero**)** **inicio** x ← x +1 v ← y + z **fin subrutina**

Qué diferencias encuentra en el valor de C?



Parámetros valor y referencia (variable)

La ejecución del algoritmo EJEMPLO utilizando:

- **PASO POR VALOR** da como resultado un valor de C = 17 dado que el valor de la variables C en el programa principal no es afectada por los cambios en el procedimiento ya que esta fue pasada por valor, y por consiguiente el procedimiento no transmite su resultado.

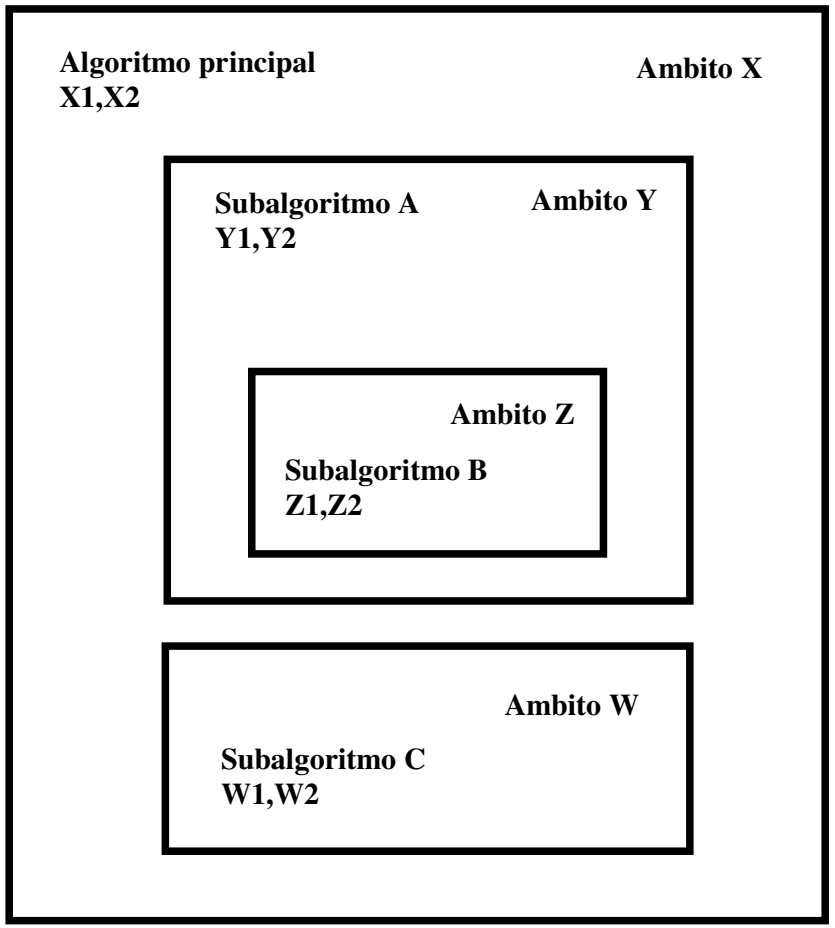
- **PASO POR REFERENCIA** el valor escrito en la pantalla del ordenador será C = 12. La primera acción que realiza el procedimiento es la asignación de los valores iniciales a las variables x, y, z, v los cuales son 3, 3, 8, 17 respectivamente haciendo la aclaración IMPORTANTE que como las variables x e y como fueron pasadas por referencia y el valor inicial es el mismo dado por el valor almacenado en A, se corresponden a la misma posición de memoria. Este hecho es fundamental a la hora de justificar el resultado del procedimiento ya que cuando de produce la asignación $x \leftarrow x + 1$ el valor de x es 4, pero también lo será el valor de y, consecuentemente la próxima acción $v \leftarrow y + z$ equivale a $12 = 4 + 8$.

ÁMBITO: VARIABLES LOCALES Y GLOBALES

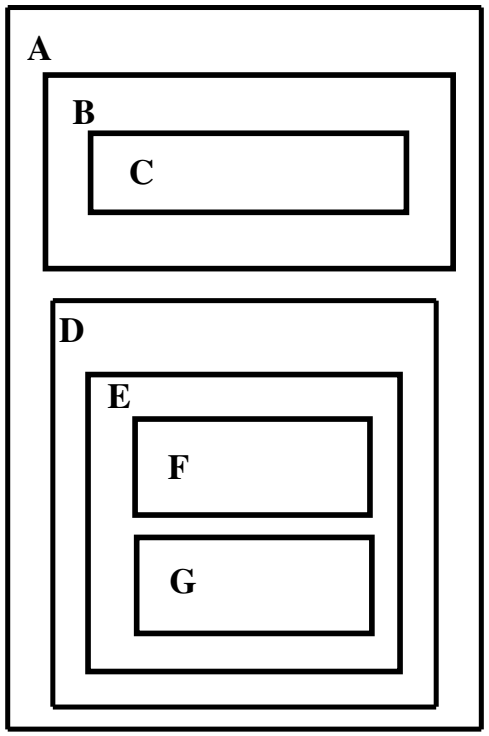
Las variables pueden clasificarse según su utilización en:

- **variables locales**: son aquellas que están declaradas dentro del algoritmo o subalgoritmo, y son propias al ámbito de la declaración, en el sentido que cada una es distinta de otra variable declarada con el mismo nombre en cualquier parte del algoritmo principal u otros subalgoritmos.

- **variables globales**: son aquellas que están declaradas en el algoritmo o subalgoritmo y son accesibles para los subalgoritmos que de él dependen.



ÁMBITO: VARIABLES LOCALES Y GLOBALES



La ventaja principal de utilizar variables locales reside en el hecho de que permite independizar al subalgoritmo del algoritmo principal, la comunicación entre el subalgoritmo y el algoritmo principal se realiza a través de la lista de parámetros. La utilización efectiva del subalgoritmo sólo requiere que el programador conozca el orden y el significado de cada parámetro a fin de construir la lista de parámetros actuales cuando se hace uso del subalgoritmo (función o procedimiento).

Las variables globales tienen la ventaja de compartir la información de diferentes subalgoritmos sin ninguna mención en la lista de parámetros del subalgoritmo.

Variables definidas en	**Accesibles desde**
A	**A,B,C,D,E,F,G**
B	**B,C**
C	**C**
D	**D,E,F,G**
E	**E,F,G**
F	**F**
G	**G**

Ejemplo

```
algoritmo Ejemplo1
variables
   entero : A,X,Y
inicio
   X ← 5
   A ← 10
   Y ← F(X)
   escribir ('X:',X, 'A:',A,'Y:',Y)
fin

Funcion F(N: entero): entero
variables
   entero : X
inicio
   A ← 5
   X ← 12
   Devolver( N+A )
Fin funcion
```

A la variable **GLOBAL** A se puede acceder desde el algoritmo y desde la función. Sin embargo, X identifica a dos variables distintas: una **LOCAL** al algoritmo y sólo se puede acceder desde él y otra **LOCAL** a la función.
Al ejecutar el algoritmo se obtendrían los siguientes resultados:

X = 5
A = 10

Y = F(5)

A = 5
X = 12
F = 5 + 5 = 10
Y = 10

Invocación a la función F, se realiza el paso del parámetro actual X al parámetro formal N (paso por valor). Se modifica el valor de A en el algoritmo principal al ser A una variable **GLOBAL**. No se modifica el valor de X en el algoritmo principal ya que X es una variable **LOCAL**

Se muestra
X: 5 A: 5 Y: 10

Instructor: Gustavo Salazar Escobar

GIT – Control de versiones de código



SISTEMA DE CONTROL DE VERSIONES (SCV)

"Un control de versiones es un sistema que registra los cambios realizados en un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante."

Recuperado de: https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones [11-11-2020]

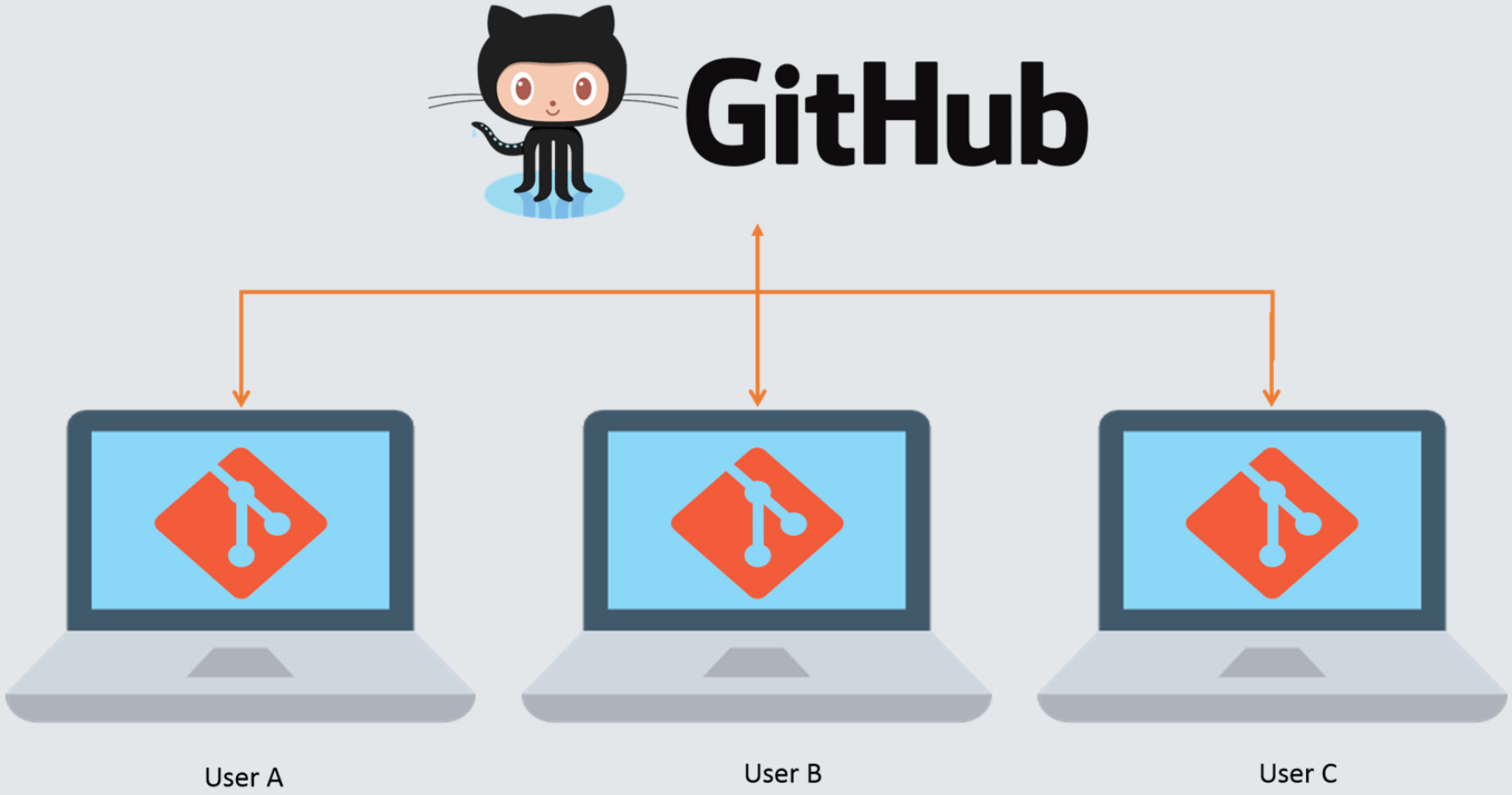
¿QUÉ ES GIT?



¿QUIÉNES USAN GIT?



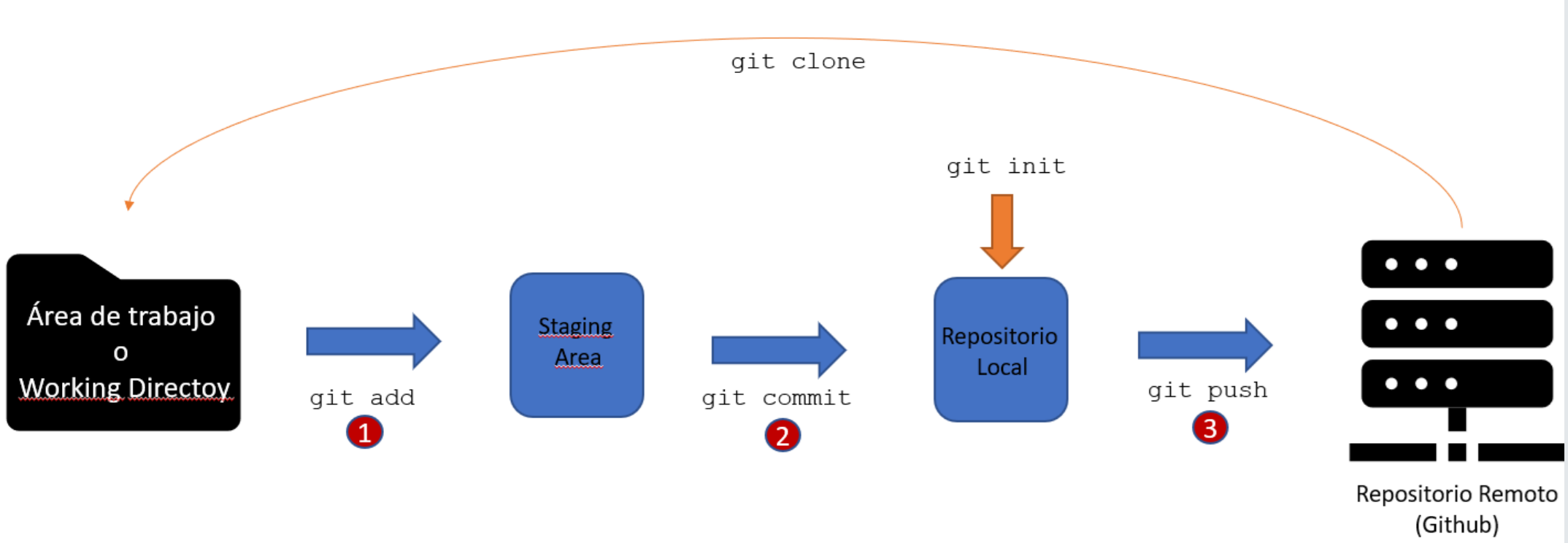
¿QUÉ ES GITHUB?



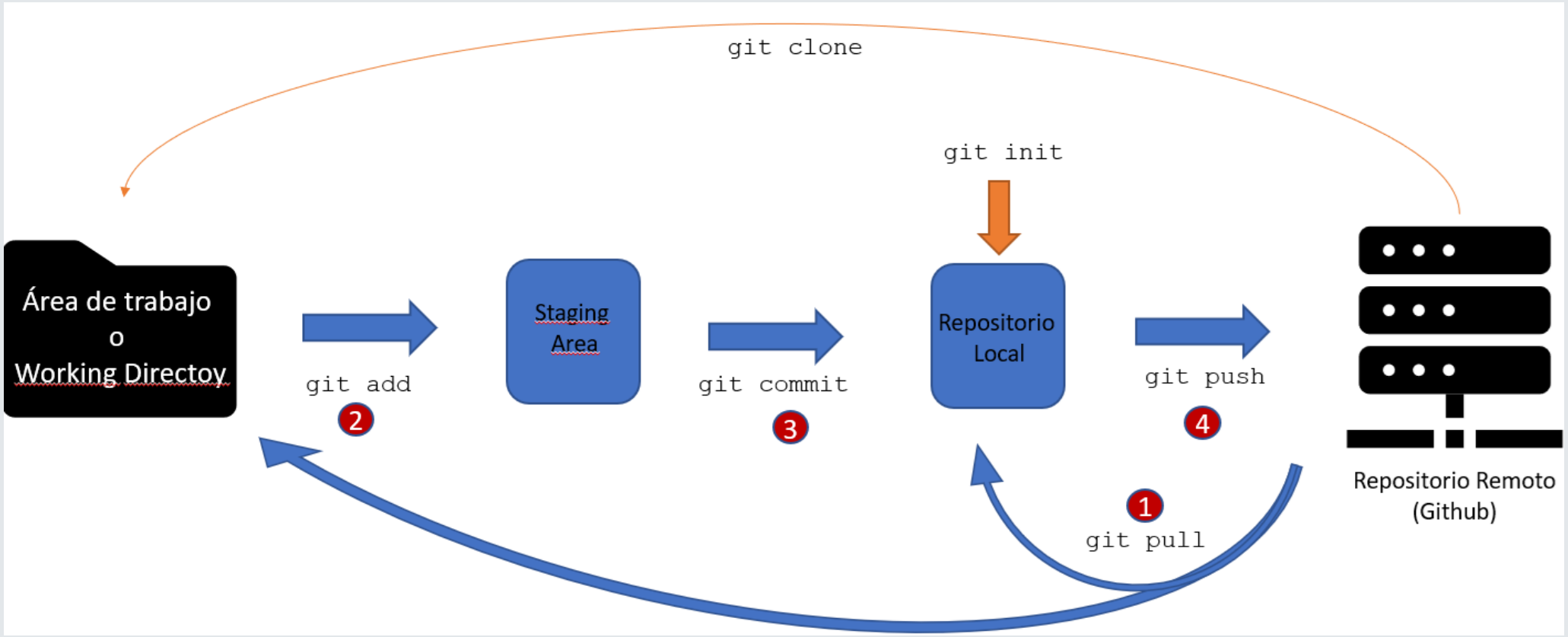
FUNCIONAMIENTO Y
COMANDOS BÁSICOS

`$ git init`	`$ git push`
`$ git clone`	`$ git pull`
`$ git add <archivo>`	`$ git status`
`$ git add .`	`$ git branch`
`$ git commit -m “mensaje”`	`$ git merge`

ORDEN DE COMANDOS PARA TRABAJAR DE FORMA INDIVIDUAL CON GIT



ORDEN DE COMANDOS PARA TRABAJAR DE MANERA COLABORATIVA CON GIT



COMANDOS BÁSICOS LINUX PARA USAR EN TERMINAL BASH DE GIT

`$ pwd`	`$ ls`
`$ cd <directorio>`	`$ clear`
`$ cd ..`	

VIDEO
TUTORIAL



DESAFÍO 1.

Crea tu primer Repositorio en GitHub para luego clonarlo

DESAFÍO 2.

Subir un Repositorio local ya existente a un Repositorio Remoto en GitHub

DESAFÍO 2.1.

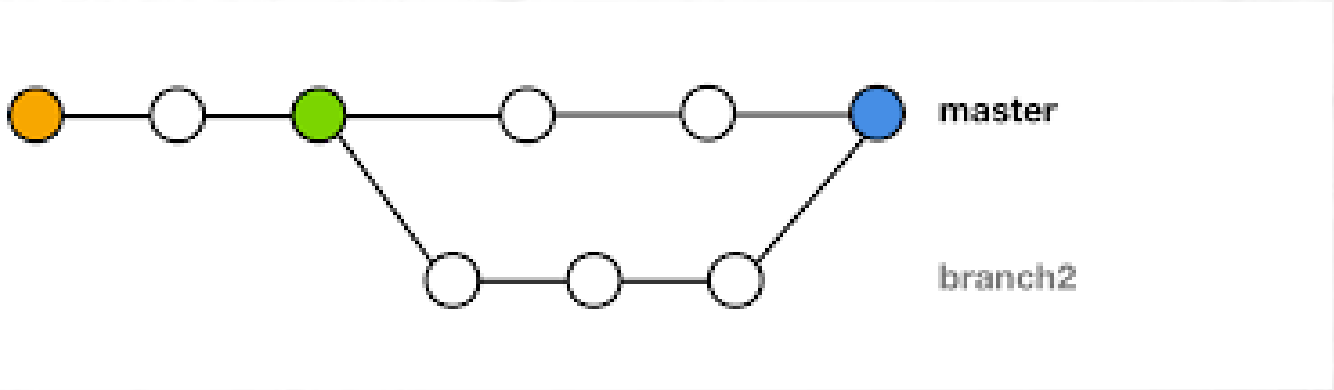
Crear archivo README.md en GitHub para luego bajar al Repositorio Local

DESAFÍO 3.

Calculadora

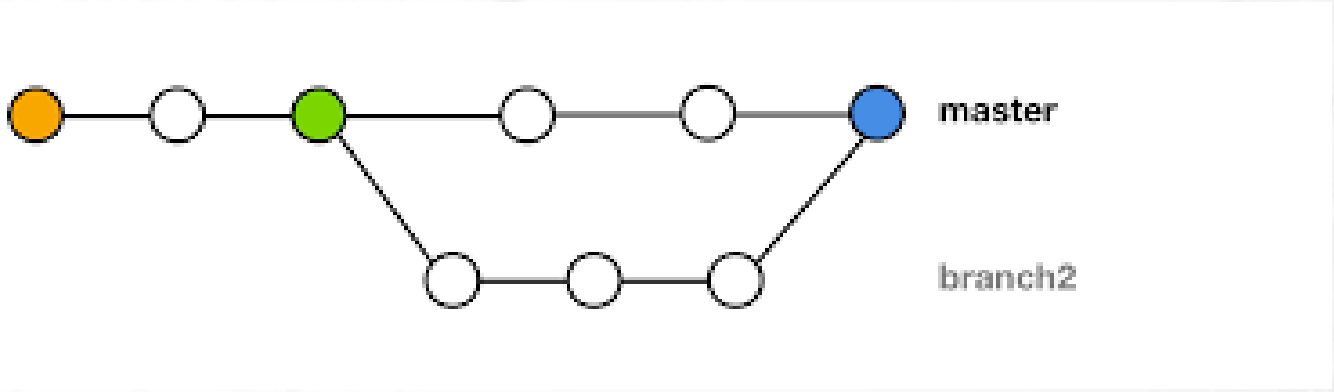
RAMAS EN GIT

En la práctica, se trata que nuestro código tenga distintos estados donde el programador pueda pasar de uno a otro o mezclar estos estados de manera ágil.

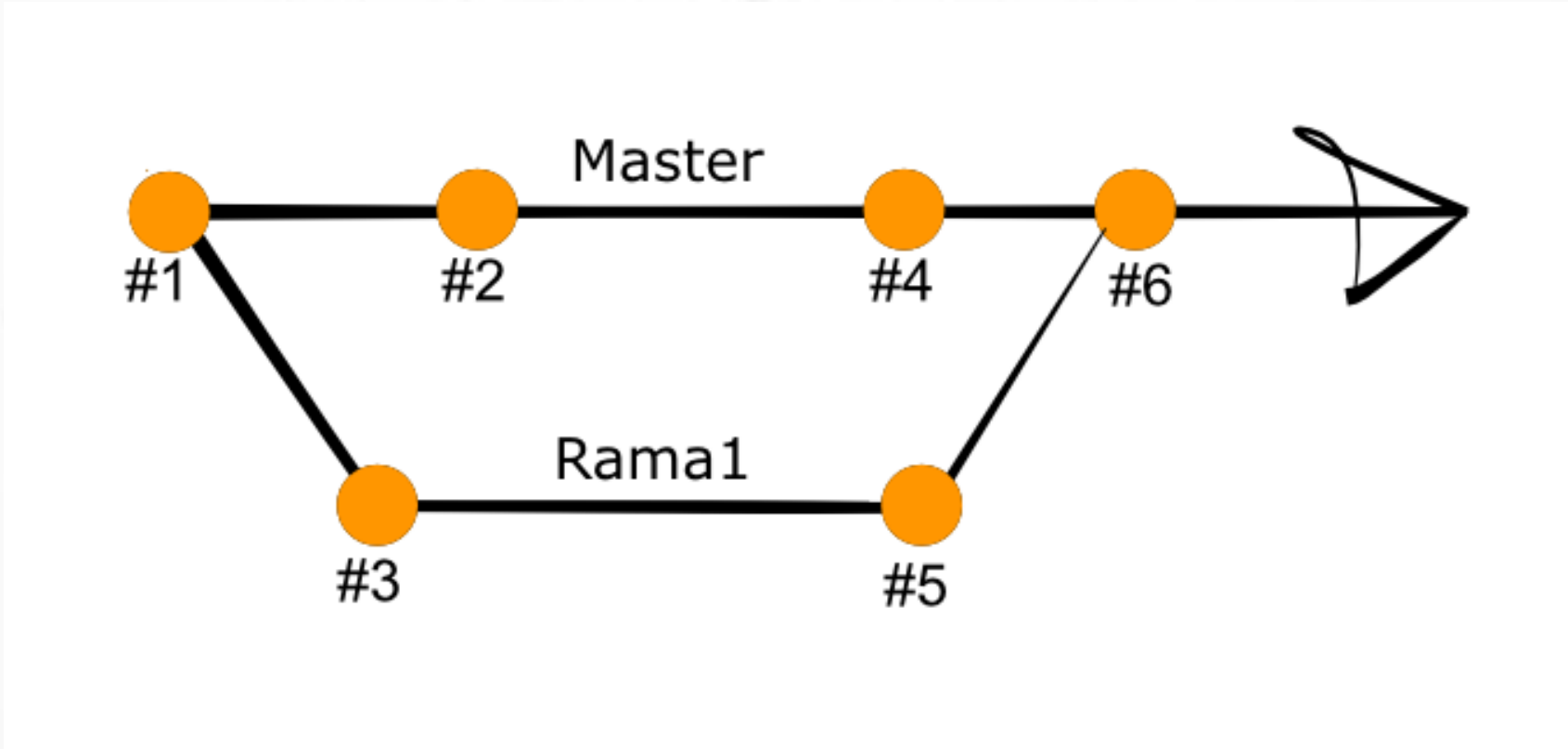


RAMA MASTER

Esta rama no es distinta a cualquier otra que se pueda crear de forma manual, sin embargo, habitualmente se utiliza para alojar el software en su última versión funcional y estable, la que se encuentra desplegada en algún servidor.



EXPLICACIÓN GRÁFICA



COMANDOS PARA TRABAJAR CON RAMAS

`$ git branch`	`$ git merge nombre-rama`
`$ git branch nombre_rama`	`$ git pull origin nombre-rama`
`$ git branch -d nombre_rama`	`$ git push origin nombre-rama`
`$ git checkout nombre_rama`	



VIDEO TUTORIAL

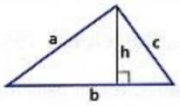
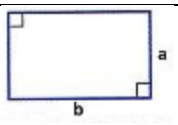
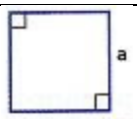


ANEXO 2

PROBLEMAS PROPUESTOS

Utilizando el lenguaje JavaScript desarrollar un programa que dé solución a los siguientes problemas:

1. Desarrollar un programa que permita calcular el área o perímetro de algunas figuras planas según la siguiente tabla:

Figura	Perímetro	Área
	$a + b + c$	$\frac{b\ x\ h}{2}$
	$2\ x\ (b + a)$	$b\ x\ a$
	$4\ x\ a$	a^2
	$2\ x\ \pi\ x\ r^2$	$\pi\ x\ r^2$

2. Desarrollar un programa que permita almacenar las edades de un grupo de 10 personas en un vector de enteros y luego determine la cantidad de personas que son menores de edad, mayores de edad, cuantos adultos mayores, la edad mas baja, la edad más alta y el promedio de edades ingresadas. Para el ejercicio anterior suponer que un adulto mayor debe tener una edad igual o superior a 60. Debe validar para cada ingreso que los valores estén en un rango entre 1 y 120 años. En caso de error deberá notificar y solicitar un nuevo valor.

3. Escriba un programa que lea dos vectores de números enteros ordenados ascendentemente y luego produzca la lista ordenada de la mezcla de los dos. Por ejemplo, si los dos arreglos tienen los números 1 3 6 9 17 y 2 4 10 17, respectivamente, la lista de números en la pantalla debe ser 1 2 3 4 6 9 10 17 17. Limite los vectores a un tamaño de 5 y debe validar en cada ingreso que realmente se estén ingresando datos de forma ascendente

4. Una emisora con presencia en diferentes ciudades, desea conocer el rating de canciones y cantantes más escuchados (sonados) en este semestre del año. Por lo tanto, se ha pedido

ANEXO 2

a estudiantes del SENA del programa de tecnólogo en análisis y desarrollo de software desarrollar una solución que permita conocer la respuesta de 6 personas con relación a sus gustos musicales. Con fines administrativos y realizar una rifa entre las personas encuestadas, la emisora desea poder registrar de las personas entrevistadas su: Nombre, número de identificación (cédula), fecha de nacimiento, correo electrónico, ciudad de residencia, ciudad de origen. Además, se deberá poder almacenar el artista y título de hasta 3 canciones favoritas en cada una de las personas que se ingrese. Teniendo en cuenta lo anterior, se sugiere que la solución deberá mostrar un menú que permita las siguientes opciones:

a. Agregar una persona con los datos que se listan anteriormente.
b. Mostrar la información personal de una persona particular por medio de su posición en el vector.