|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | Karina García Morales |
| *Asignatura:* | Fundamentos de programación |
| *Grupo:* | 19 |
| *No de Práctica(s):* | Práctica No. 3 |
| *Integrante(s):* | Alonso Agustín Alejandra |
| *No. de Equipo de cómputo empleado:* | Equipo personal. |
| *No. de Lista o Brigada:* | 2 |
| *Semestre:* | 2021-2 |
| *Fecha de entrega:* | 08/06/2021 |
| *Observaciones:* |  |

CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

**Guía práctica de estudio 03: Solución de problemas y Algoritmos.**

**Objetivos:**

Elaborar algoritmos correctos y eficientes en la solución de problemas siguiendo las etapas de Análisis y Diseño pertenecientes al Ciclo de vida del software.

**Actividades:**

* A partir del enunciado de un problema, identificar el conjunto de entrada y el conjunto de salida.
* Elaborar un algoritmo que resuelva un problema determinado (dado por el profesor), identificando los módulos de entrada, de procesamiento y de salida.

**Desarrollo:**

Un problema informático se puede definir como el conjunto de instancias al cual corresponde un conjunto de soluciones, junto con una relación que asocia para cada instancia del problema un subconjunto de soluciones (posiblemente vacío).

Para poder solucionar un problema nos apoyamos en la Ingeniería de Software que de acuerdo a la IEEE se define como “La aplicación de un enfoque sistemático, disciplinado y cuantificable hacia el desarrollo, operación y mantenimiento del software". Por lo que el uso y establecimiento de principios de ingeniería sólidos, son básicos para obtener un software que sea económicamente fiable y funcione eficientemente.

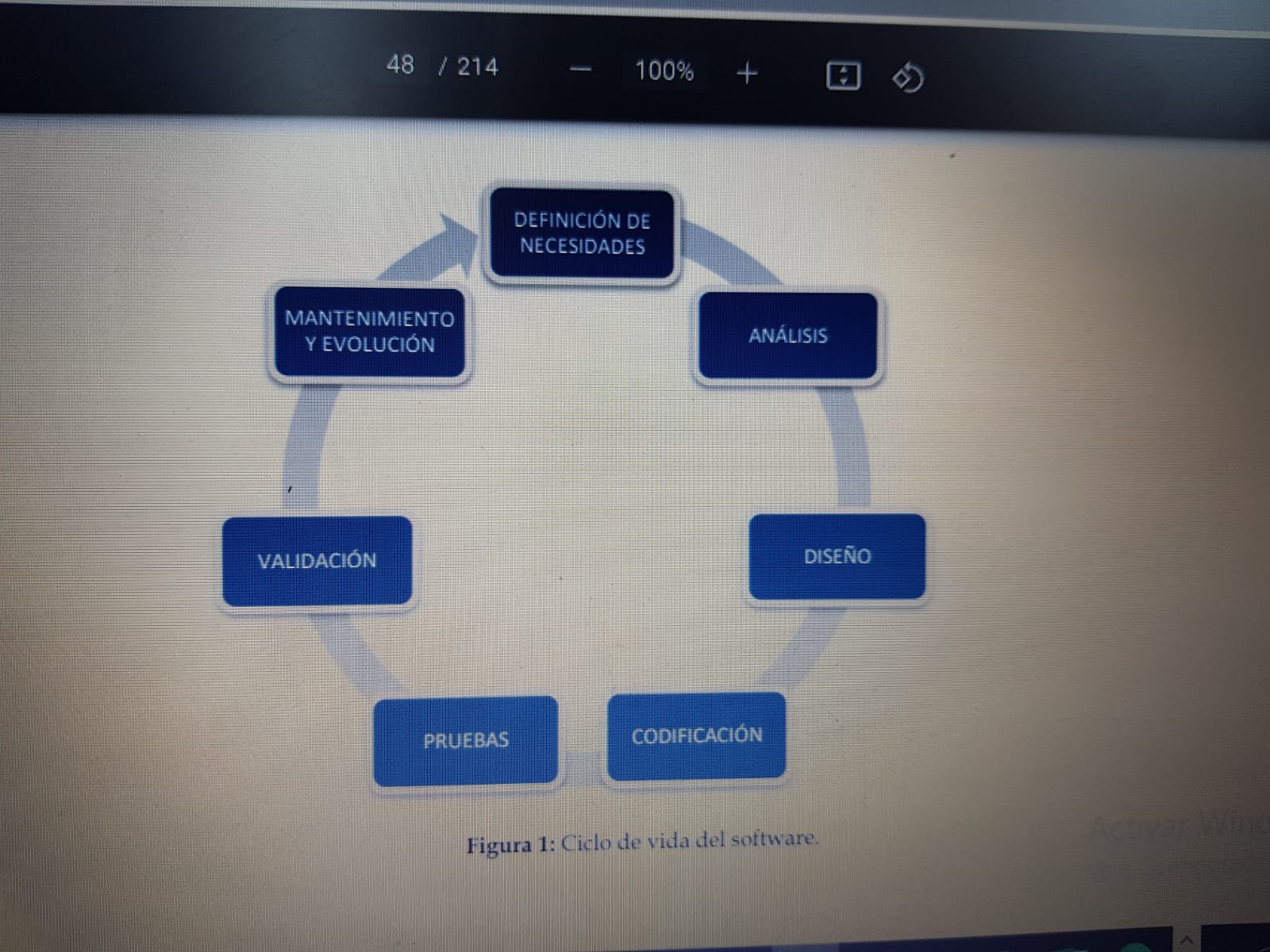
La Ingeniería de Software provee métodos que indican cómo generar software. Estos métodos abarcan una amplia gama de tareas:

* Planeación y estimación del proyecto.
* Análisis de requerimientos del sistema y software.
* Diseño de la estructura de datos, la arquitectura del programa y el procedimiento algorítmico.
* Codificación.
* Pruebas y mantenimiento (validación y verificación).

**Ciclo de vida del software**

La ISO en su norma 12207 define al ciclo de vida de un software como:

Un marco de referencia que contiene las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando desde la definición hasta la finalización de su uso.



*Figura 1. Ciclo de la vida del software. Obtenido de: MADO de prácticas.*

**Solución de problemas**

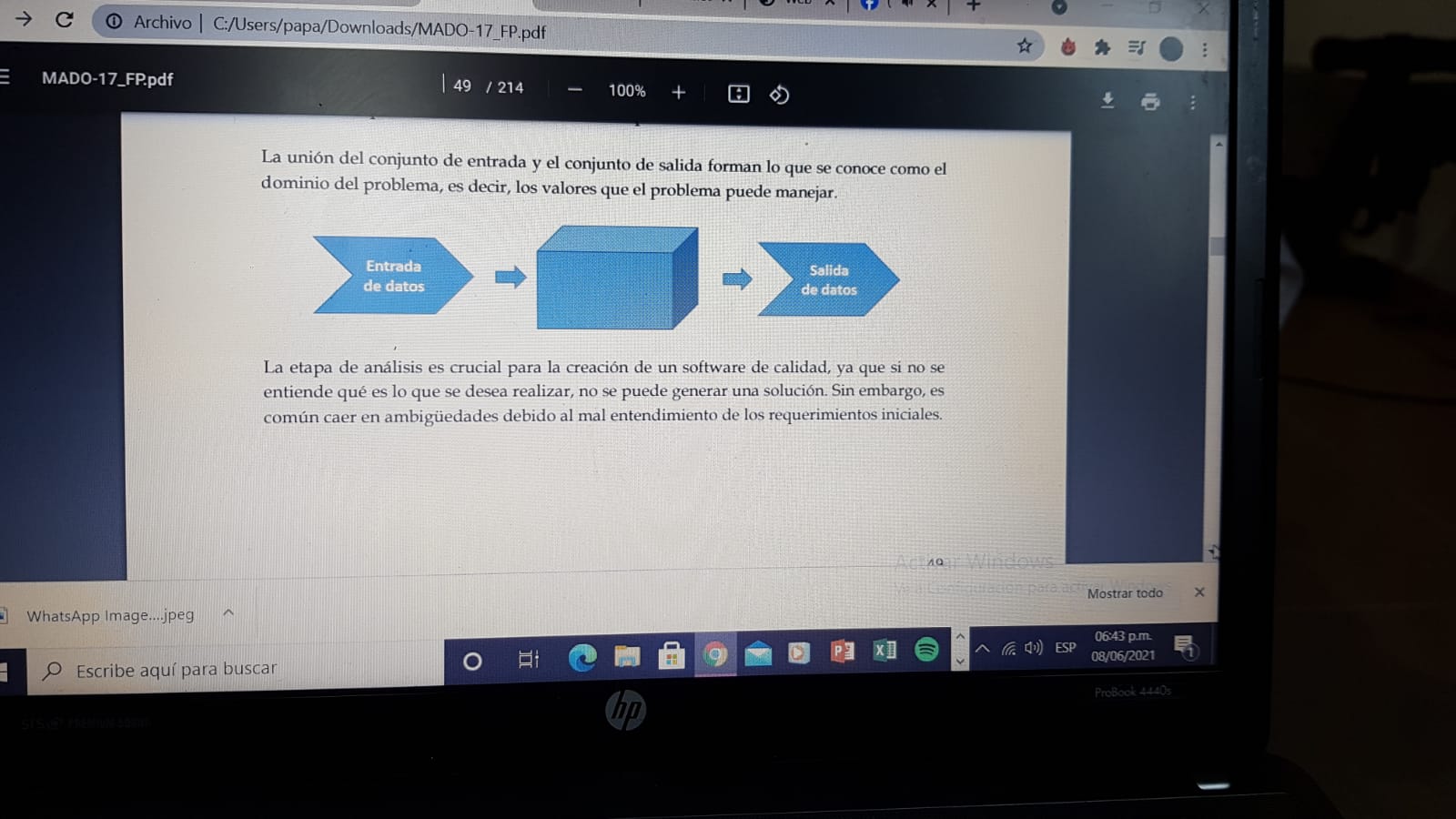
El análisis es el proceso para averiguar qué es lo que requiere el usuario del sistema de software (análisis de requisitos). Esta etapa permite definir las necesidades de forma clara y concisa (especificación de requisitos).

Por lo tanto, la etapa del análisis consiste en conocer qué es lo que está solicitando el usuario. Para ello es importante identificar dos grandes conjuntos dentro del sistema: el conjunto de entrada y el conjunto de salida.

El *conjunto* de entrada está compuesto por todos aquellos datos que pueden alimentar al sistema.

El *conjunto* de salida está compuesto por todos los datos que el sistema regresará como resultado del proceso. Estos datos se obtienen a partir de los datos de entrada.

La unión del conjunto de entrada y el conjunto de salida forman lo que se conoce como el dominio del problema, es decir, los valores que el problema puede manejar.



*Figura 2. Conjuntos dentro del sistema. Obtenido de: MADO de prácticas.*

La etapa de análisis es crucial para la creación de un software de calidad, ya que si no se entiende qué es lo que se desea realizar, no se puede generar una solución. Sin embargo, es común caer en ambigüedades debido al mal entendimiento de los requerimientos iniciales.

**Algoritmos**

Una vez realizado el análisis, es decir, ya que se entendió qué es lo que está solicitando el usuario y ya identificado el conjunto de entrada y el conjunto de salida, se puede proceder al diseño de la solución, esto es, a la generación del algoritmo.

Dentro del ciclo de vida del software, la creación de un algoritmo se encuentra en la etapa de diseño.

Durante el diseño se busca proponer una o varias alternativas viables para dar solución al problema y con base en esto tomar la mejor decisión para iniciar la construcción.

Un problema matemático es computable si éste puede ser resuelto, en principio, por un dispositivo computacional.

La teoría de la computabilidad es la parte de la computación que estudia los problemas de decisión que pueden ser resueltos con un algoritmo.

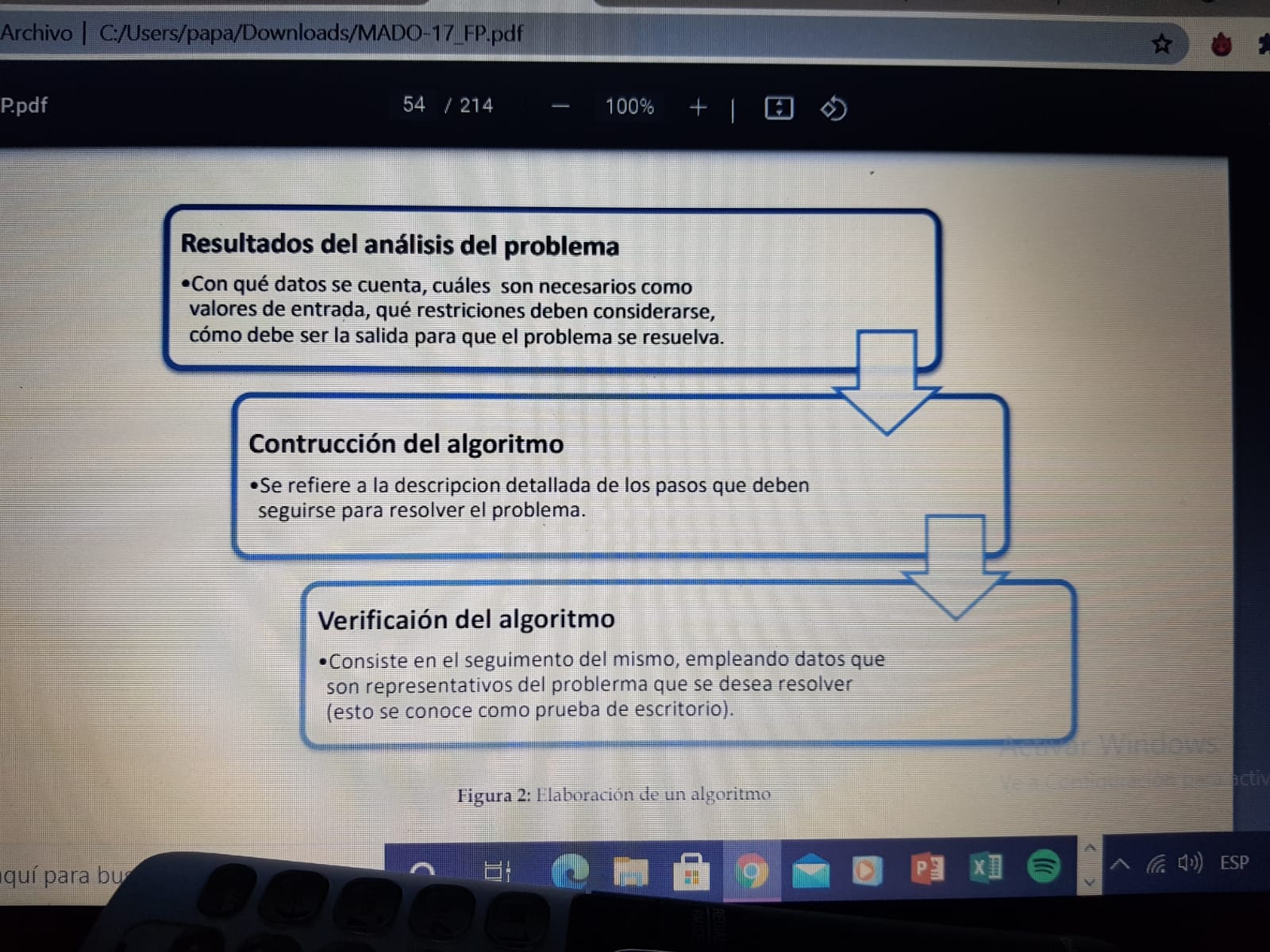
Un algoritmo se define como un conjunto de reglas, expresadas en un lenguaje específico, para realizar alguna tarea en general, es decir, un conjunto de pasos, procedimientos o acciones que permiten alcanzar un resultado o resolver un problema. Estas reglas o pasos pueden ser aplicados un número ilimitado de veces sobre una situación particular.

Un algoritmo es la parte más importante y durable de las ciencias de la computación debido a que éste puede ser creado de manera independiente tanto del lenguaje como de las características físicas del equipo que lo va a ejecutar.

Las principales características con las que debe cumplir un algoritmo son:

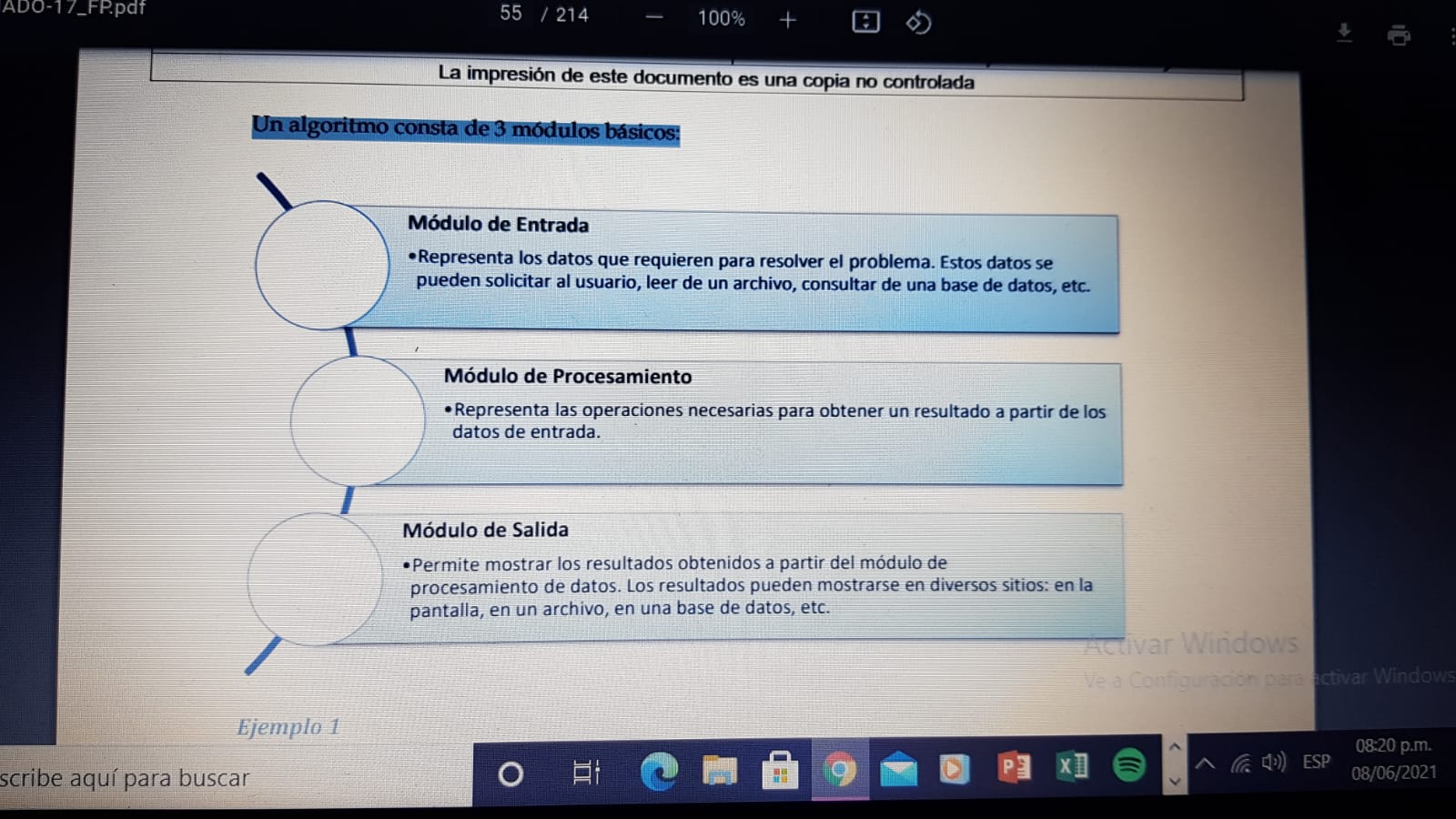
* Preciso: Debe indicar el orden de realización de paso y no puede tener ambigüedad
* Definido: Si se sigue dos veces o más se obtiene el mismo resultado.
* Finito: Tiene fin, es decir tiene un número determinado de pasos.
* Correcto: Cumplir con el objetivo.
* Debe tener al menos una salida y esta debe de ser perceptible
* Debe ser sencillo y legible
* Eficiente: Realizarlo en el menor tiempo posible
* Eficaz: Que produzca el efecto esperado

Por tanto, un buen algoritmo debe ser correcto (cumplir con el objetivo) y eficiente (realizarlo en el menor tiempo posible), además de ser entendible para cualquier persona. Las actividades a realizar en la elaboración de un algoritmo para obtener una solución a un problema de forma correcta y eficiente se muestran en la figura 3:



*Figura 3. Obtenido de: MADO de prácticas.*

Un algoritmo consta de 3 módulos básicos:



*Figura 4. Obtenido de: MADO de prácticas.*

*Ejemplo 1*

PROBLEMA: Determinar si un número dado es positivo o negativo.

RESTRICCIONES: El número no puede ser cero.

DATOS DE ENTRADA: Número real.

DATOS DE SALIDA: La validación de si el número es positivo

DOMINIO: Todos los número reales.

SOLUCIÓN:

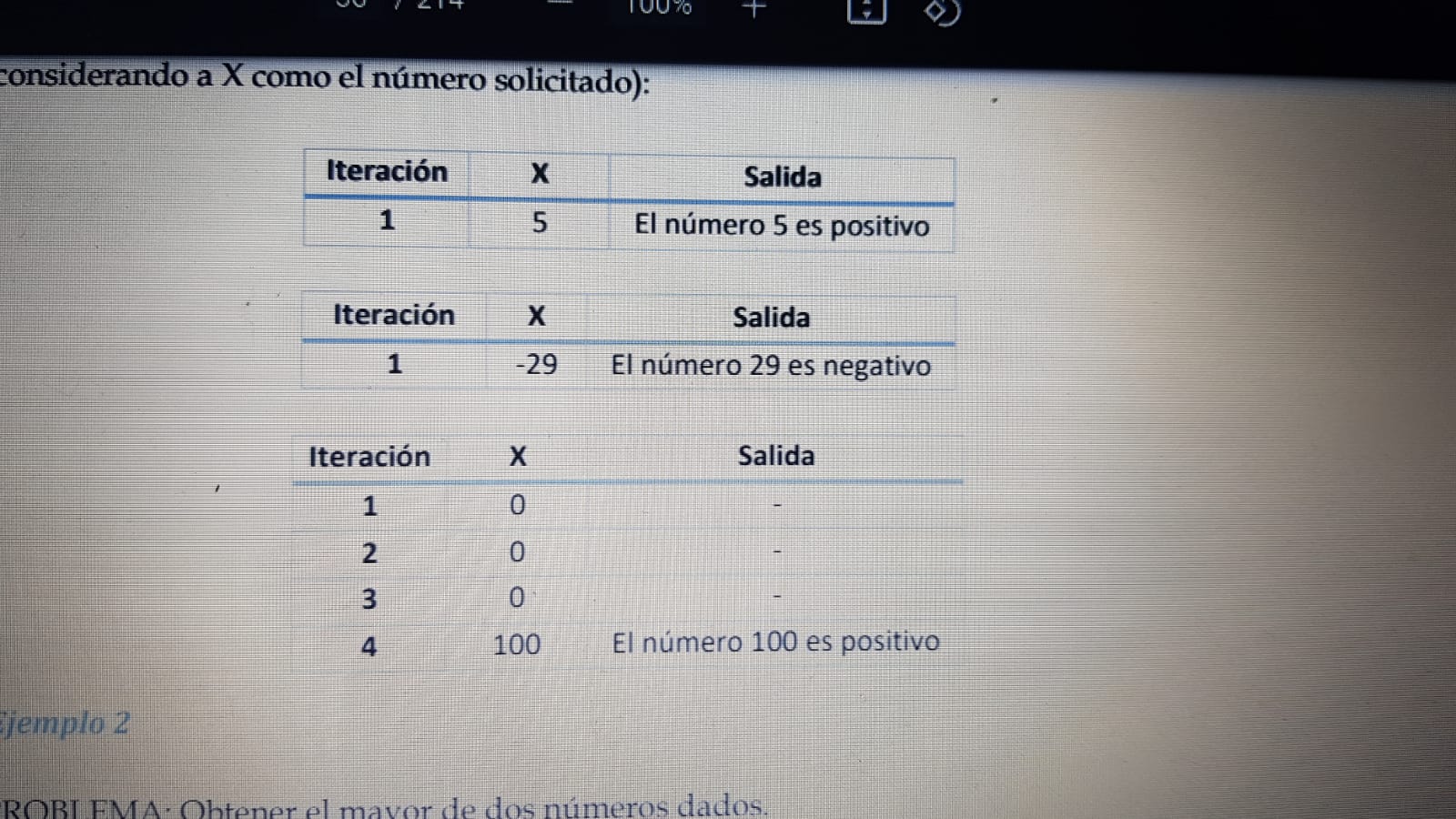
1. *Solicitar un número real.*
2. *Si el número ingresado es cero, se regresa al punto 1.*
3. *Si el número ingresado es diferente de cero, se validan las siguientes condiciones:*
   1. *Si el número ingresado es mayor a 0 se puede afirmar que el número es positivo.*
   2. *Si el número ingresado es menor a 0 se puede afirmar que el número es negativo.*

Prueba de escritorio

El diseño de la solución de un problema implica la creación del algoritmo y la validación del mismo. La validación se suele realizar mediante una prueba de escritorio.

Una prueba de escritorio es una matriz formada por los valores que van adquiriendo cada una de las variables del programa en cada iteración. Una iteración es el número de veces que se ejecuta un código y permite ver los valores que van adquiriendo las variables en cada repetición.

Para el ejemplo en cuestión la prueba de escritorio quedaría de la siguiente manera (considerando a X como el número solicitado):



*Figura 5. Obtenido de: MADO de prácticas.*

***Anexo ejemplo de algoritmo propuesto por la alumna:***

Ejemplo 1:

Problema: Determinar si un número dado es positivo o negativo.

Datos de entrada: Número real.

Restricciones: El número no puede ser cero.

Datos de salida: La validación de si el número es positivo

Construcción de algoritmo para determinar si un número dado es positivo o negativo

1. Inicio
2. Leer el número real
3. Si el número ingresado es cero, ir al paso 1.
4. Si el número ingresado es diferente de cero, ir al paso 5.
5. Si el número ingresado es mayor a cero, el número es positivo, ir a paso 7. En caso contrario ir a paso 6.
6. Si el numero ingresado es menor a cero, el número es negativo, ir a paso 7
7. Fin.

**Pruebas de escritorio:**

|  |  |  |
| --- | --- | --- |
| Intentos | X (número real) | Resultado |
| 1 | 0 | -- |
| 2 | -1 | Negativo |
| 3 | 8 | Positivo |

Ejemplo 2:

PROBLEMA: Obtener el mayor de dos números dados.

RESTRICCIONES: Los números de entrada deben ser diferentes.

DATOS DE ENTRADA: Número real.

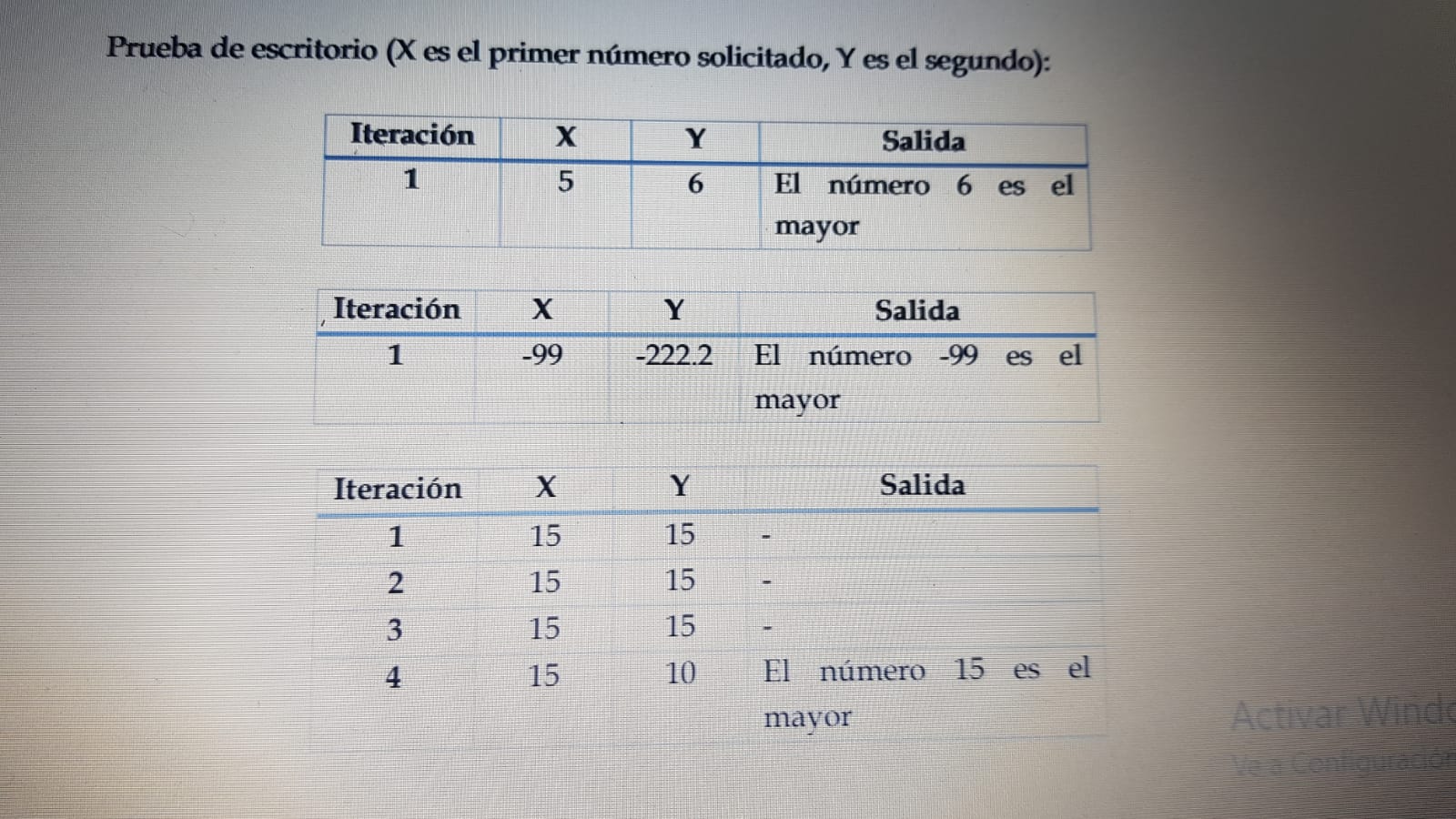
DATOS DE SALIDA: La impresión del número más grande.

DOMINIO: Todos los número reales.

SOLUCIÓN:

1. *Solicitar un primer número real.*
2. *Solicitar un segundo número real.*
3. *Si el segundo número real es igual al primer número real, se regresa al punto 2.*
4. *Si el segundo número real es diferente al primer número real, se validan las siguientes condiciones:*
   1. *Si se cumple con la condición de que el primer número es mayor al segundo número, entonces se puede afirmar que el primer número es el mayor de los números.*
   2. *Si se cumple con la condición de que el segundo número es mayor al primer número, entonces se puede afirmar que el segundo número es el mayor de los números.*

Prueba de escritorio (X es el primer número solicitado, Y es el segundo):



*Figura 6. Obtenido de: MADO de prácticas.*

***Anexo ejemplo de algoritmo propuesto por la alumna:***

Ejemplo 2

Problema: Obtener el mayor de dos números dados

Datos de entrada: Número real.

Restricciones: Los números de entrada deben ser diferentes.

Datos de salida: La impresión del número más grande.

Algoritmo para obtener el mayor de dos números reales.

1. Inicio
2. Solicitar dos números reales, ir a paso 2.
3. Leer el primer número, ir al paso 3
4. Leer el segundo número, ir al paso 5
5. Si el segundo número es igual al primer número, ir al paso 4, en caso contrario ir al paso 6.
6. Si el segundo número cumple con la condición de ser mayor, entonces se puede afirmar que el segundo número es el mayor de los números. Ir a paso 8.
7. Si el primer número cumple con la condición de ser mayor al segundo número, entonces se puede afirmar que el primer número es el mayor de los números. Ir a paso 8.
8. Fin

Pruebas de escritorio:

|  |  |  |  |
| --- | --- | --- | --- |
| Intentos | A (primero número) | B (segundo número) | Resultado |
| 1 | 19 | 6 | 19 > 6 |
| 2 | 1 | 1 | -- |
| 3 | 4 | 78 | 4 < 78 |

Ejemplo 3:

PROBLEMA: Obtener el factorial de un número dado. El factorial de un número está dado por el producto de ese número por cada uno de los números anteriores hasta llegar a 1. El factorial de 0 (0!) es 1.

RESTRICCIONES: El número de entrada debe ser entero y no puede ser negativo.

DATOS DE ENTRADA: Número entero.

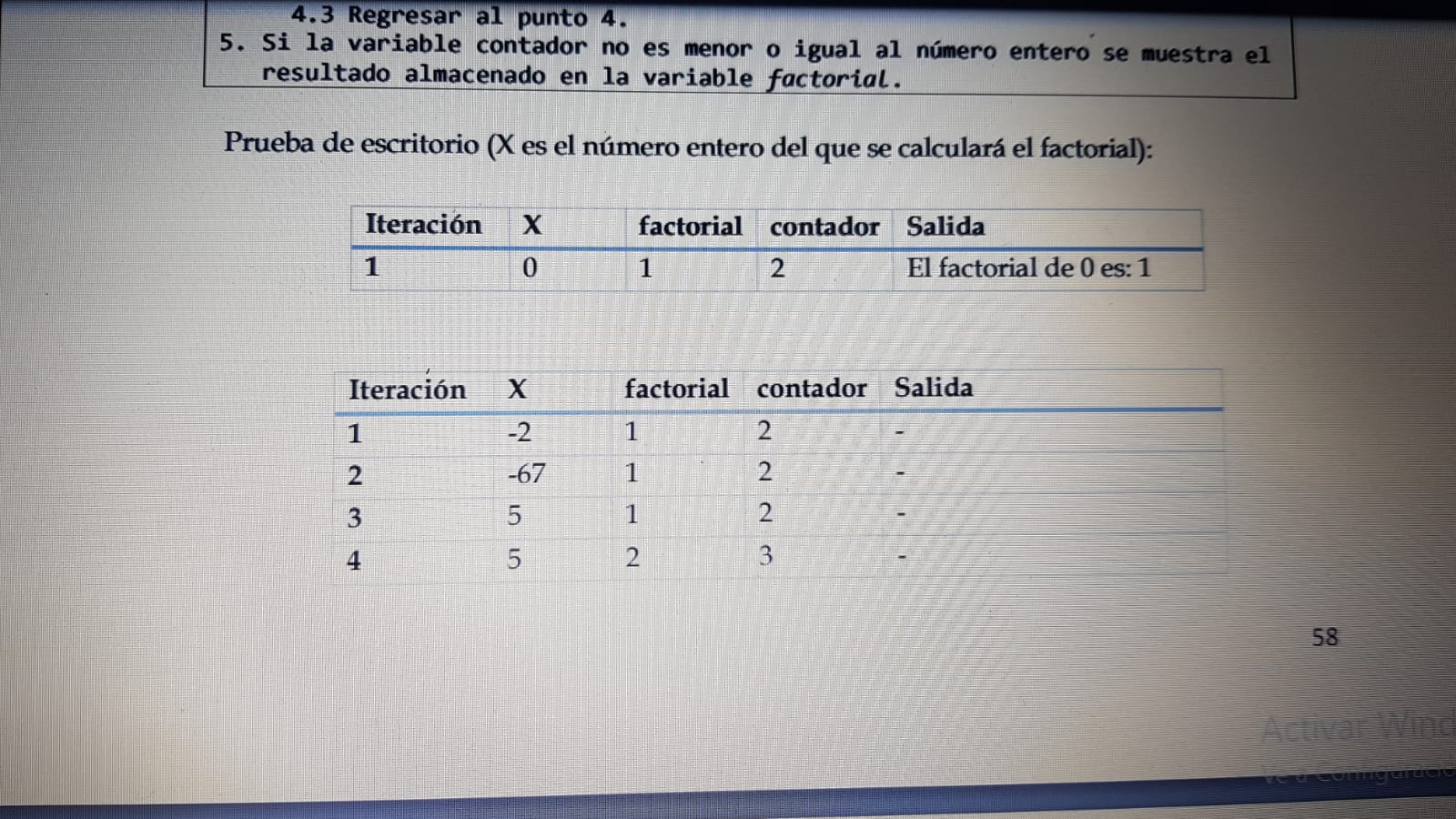
DATOS DE SALIDA: La impresión del factorial del número.

DOMINIO: Todos los números naturales positivos.

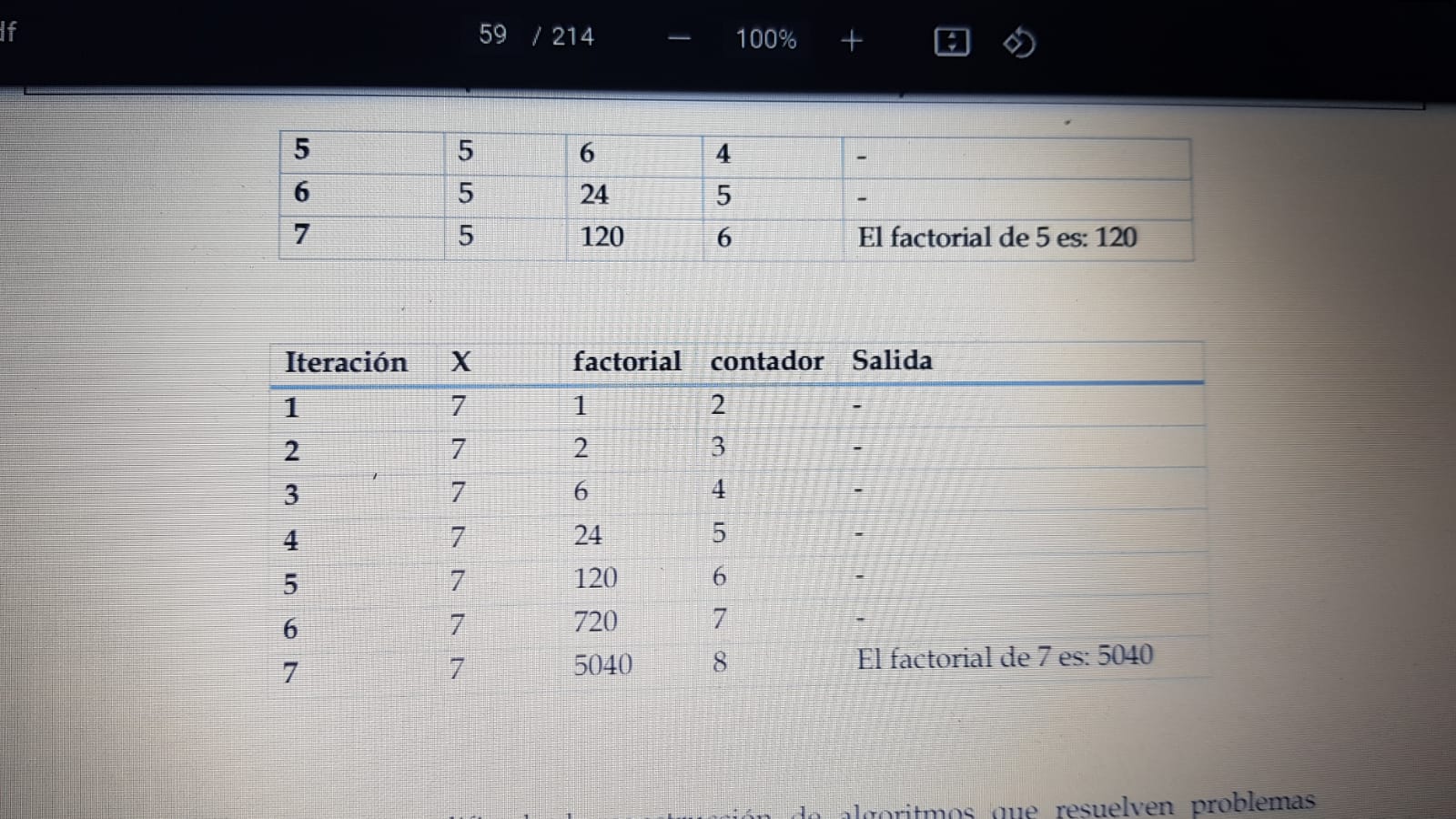
SOLUCIÓN:

1. *Solicitar un número entero.*
2. *Si el número entero es menor a cero regresar al punto 1.*
3. *Si el número entero es mayor a cero se crea una variable entera contador que inicie en 2 y una variable entera factorial que inicie en uno.*
4. *Si la variable contador es menor o igual al número entero de entrada se realiza lo siguiente:*
   1. *Se multiplica el valor del variable contador con el valor de la variable factorial. El resultado se almacena en la variable factorial.*
   2. *Se incrementa en uno el valor de la variable contador.*
   3. *Regresar al punto 4.*
5. *Si la variable contador no es menor o igual al número entero se muestra el resultado almacenado en la variable factorial.*

Prueba de escritorio (X es el número entero del que se calculará el factorial):



*Figura 7. Obtenido de: MADO de prácticas.*



*Figura 8. Obtenido de: MADO de prácticas.*

***Anexo ejemplo de algoritmo propuesto por la alumna:***

Ejemplo 3

Problema: Obtener el factorial de un número dado. La factorial de un número está dado por el producto de ese número por cada uno de los números anteriores hasta llegar a 1. El factorial de 0 (0!) es 1.

Datos de entrada: Número entero positivo.

Restricciones: El número de entrada debe ser entero y no puede ser negativo.

Datos de salida: La impresión de la factorial del número

Algoritmo para obtener la factorial de un número dado.

1. Inicio
2. Solicitar un número entero, ir al paso 3.
3. Leer el número entero, ir al paso 4
4. Si el número es menor a cero, ir al paso 2, en caso contrario, ir al paso 5.
5. Crear una variable bajo el nombre “contador” que inicie en dos y una variable entera bajo el nombre “factorial”, que inicie en uno.
6. Leer variable “contador”, ir al paso 7.
7. Si la variable contador en menor o igual al número entero, ir al paso 8, en caso contrario ir al paso 11.
8. Multiplicar la variable “contador” con la variable “factorial”, el resultado lo almacenamos en la variable “factorial”
9. La variable “contador crece a razón de 1. Ir al paso 10.
10. Leer resultado, ir al paso 7.
11. Si la variable “contador” no es menor o igual al número entero se muestra el resultado almacenado en la variable factorial. Ir al paso 12.
12. Se obtuvo la factorial. Ir al paso 13.
13. Fin.

Pruebas de escritorio:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Intentos. | X (número entero) | Factorial | Contador | Resultado |
| 1 | -123 | 1 | 2 | -- |
| 2 | 3 | 1 | 2 | -- |
| 3 | 3 | 2 | 3 | -- |
| 4 | 3 | 6 | 4 | El factorial de 3 es: 6 |

***Ejercicio 1***

PROBLEMA: Seguir el algoritmo para obtener una figura

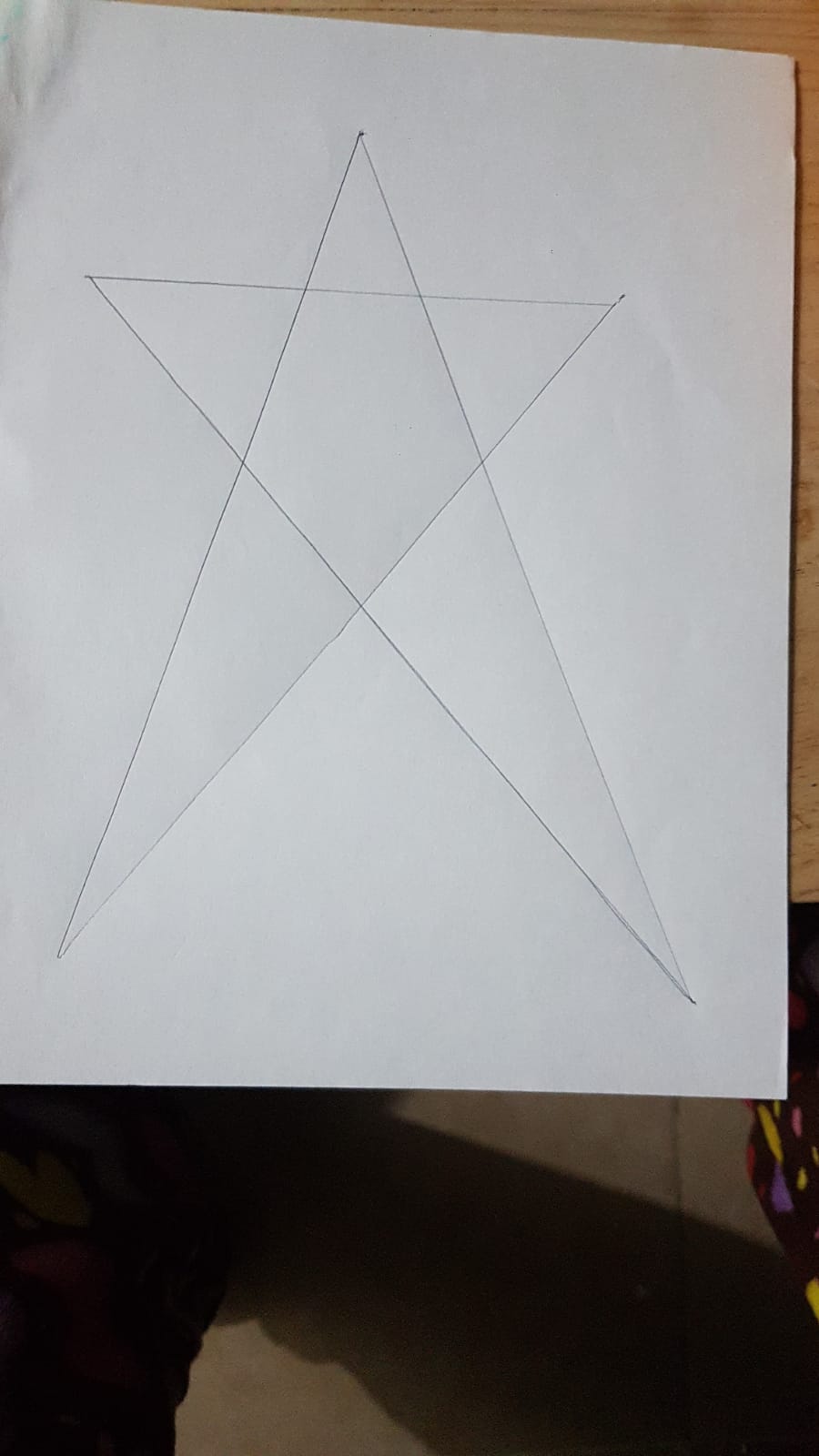
ENTRADA: Hoja tamaño carta en limpio, regla y lápiz.

SALIDA: Figura correcta.

Algoritmo

1. Dibuja una V invertida. Empieza desde el lado izquierdo, sube, y baja hacia el lado derecho, no levantes el lápiz.
2. Ahora dibuja una línea en ángulo ascendente hacia la izquierda. Debe cruzar la primera línea más o menos a 1/3 de la altura. Todavía no levantes el lápiz del papel.
3. Ahora, dibuja una línea horizontal hacia la derecha. Debe cruzar la V invertida más o menos a 2/3 de la altura total. Sigue sin levantar el lápiz.
4. Dibuja una línea en un ángulo descendente hasta el punto de inicio. Las líneas deben unirse.
5. Ahora ya puedes levantar el lápiz del papel. Has terminado la estrella de 5 puntas

*Salida:*

**

*Figura 9. Resultado al aplicar el algoritmo.*

***Ejercicio 2***

PROBLEMA: Seguir el algoritmo para obtener una figura

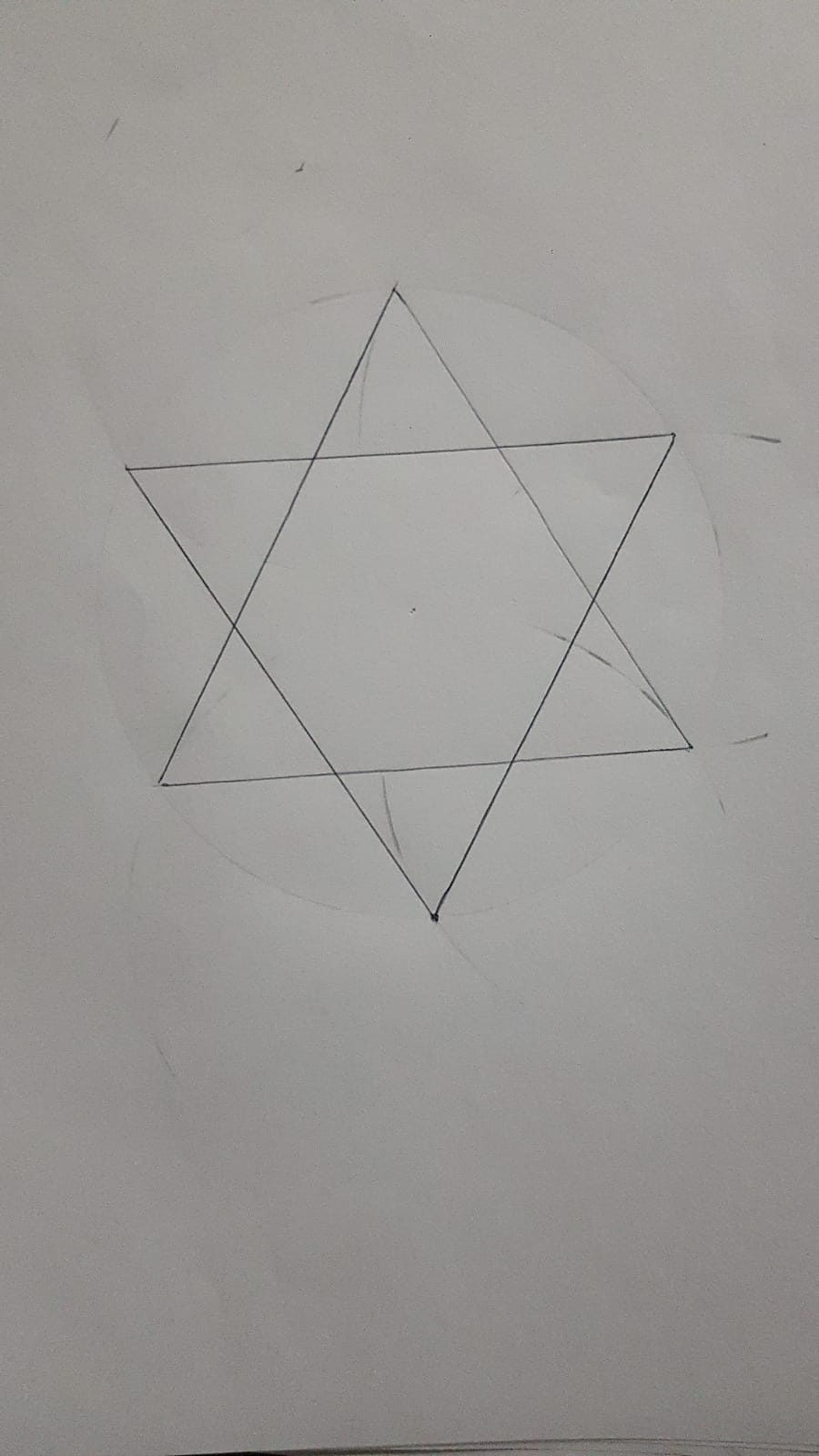
ENTRADA: Hoja tamaño carta en limpio, regla y lápiz.

SALIDA: Figura correcta.

Algoritmo

1. Empieza dibujando un círculo con un compás. Coloca un lápiz en el compás.Coloca la punta del compás en el centro de una hoja de papel.
2. Ahora gira el compás, mientras mantienes la punta apoyada en el papel. El lápiz dibujará un círculo perfecto alrededor de la punta del compás.
3. Marca un punto en la parte superior del círculo con el lápiz. Ahora, coloca la punta del compás en la marca. No cambies el radio del compás con que hiciste el círculo.
4. Gira el compás para hacer una marca en el propio círculo hacia la izquierda. Haz una marca también en el lado derecho.
5. Ahora, coloca la punta del compás en uno de los puntos. Recuerda no cambiar el radio del compás. Haz otra marca en el círculo.
6. Continúa moviendo la punta del compás a las otras marcas, y continúa hasta que tengas 6 marcas a la misma distancia unas de otras. Ahora, ya puedes dejar tu compás a un lado.
7. Usa una regla para crear un triángulo que empiece en la marca superior del círculo. Coloca el lápiz en la marca superior. Ahora dibuja una línea hasta la segunda marca por la izquierda. Dibuja otra línea, ahora hacia la derecha, saltándote la marca de la parte más baja. Complementa el triángulo con una línea hacia la marca superior. Así completarás el triángulo.
8. Crea un segundo triángulo empezando en la marca en la base del círculo. Coloca el lápiz en la marca inferior. Ahora conéctala con la segunda marca hacia la izquierda. Dibuja una línea recta hacia la derecha, saltándote el punto superior. Completa el segundo triángulo dibujando una línea hasta la marca en la parte inferior.
9. Borra el círculo. Has terminado de dibujar tu estrella de 6 puntos.

*Salida:*

**

*Figura 10. Resultado al aplicar el algoritmo.*

***Observaciones:***

1. El primer algoritmo fue el que más ambigüedades presento a la hora de su aplicación.
2. Para poder seguir los pasos del algoritmo, me fue muy necesaria la intuición, pues no especificaba en que momento tomar la regla, pasar de una parte a otra, no era tan fácil de comprender.
3. El primer algoritmo fue el que tuvo más “escases” de pasos para su aplicación.
4. El segundo algoritmo fue más específico con sus pasos, el momento en que se tenía que utilizar cada uno de los materiales pedidos y el más complejo en cuanto a su contenido.

**Ejercicios de tarea**

1. Describe las salidas de los ejercicios finales y la comparación entre ambos algoritmos.

Al parecer cada uno de los resultados fueron satisfactorios, sin embargo hubieron varias complicaciones a lo largo de su aplicación, en el caso del algoritmo número uno, fue el más difícil de seguir pues sus instrucciones no eran lo suficientemente específicas como en el caso del algoritmo dos, el cual se encontraba con mayor complejidad, tanto en la redacción como en y estructura.

1. ¿Qué se necesita para dar solución a un problema?

“En primer lugar, tienes que **definir el problema**. A continuación, tienes que **identificar varias opciones** de soluciones. ¿Cómo puede ser solucionado?

Después, **evalúa tus opciones** ¿Cuál es la mejor opción para solucionar el problema? ¿Cuál es la opción más sencilla? ¿Cómo debes priorizar?

Finalmente, **aplica la solución elegida**. ¿Se ha solucionado el problema? ¿Hay otra opción que debas probar?, ser específicos al momento de querer dar solución al problema.”

1. Describe las fases del ciclo de vida del desarrollo de software explicadas en clase e ilustra

* Planeación y estimación del proyecto.

En esta etapa observamos que es lo que queremos llegar a resolver

* Análisis de requerimientos del sistema y software.

Se realiza el entendimiento y el análisis detallado sobre cuál es el problema a tratar, cuales son nuestros datos de entrada, restricciones y todo lo que se requiere para diseñar la resolución del problema.

* Diseño de la estructura de datos, la arquitectura del programa y el procedimiento algorítmico.

Se lleva a cabo la generación del diagrama de flujo, el pseudocódigo

* Codificación.

Se realiza la codificación

* Pruebas y mantenimiento (validación y verificación).

Verificación: cuando se verifica que el sistema funcione en su totalidad

Validación: Validar que se puede realizar lo que el usuario desea. Se pueden aplicar mejoras.

1. Analizar las siguientes problemáticas, desglosando y dando solución al problema identificando claramente su conjunto de entrada (datos de entrada), el conjunto de salida (datos de salida) y restricciones, si es que tiene.

En una playa el estacionamiento cobran $ 2.5 por hora o fracción. Determinar cuánto debe pagar un cliente por el estacionamiento de su vehículo, conociendo el tiempo de estacionamiento en horas y minutos.

Problema: Determinar cuánto debe pagar un cliente por el estacionamiento de su vehículo, conociendo el tiempo de estacionamiento en horas y minutos.

Datos de entrada: El cobro por $2.5

1min = $0.04167

T = tiempo

H= T/60

P= H\*2.5

Ps=m\*0.04167

PT= P+Ps

Restricciones: Cobro por hora o fracción, T>0

Datos de salida: Cantidad a pagar

Algoritmo para obtener la cantidad a pagar

1. Inicio
2. Leer cantidad de tiempo en el estacionamiento, ir a paso 3
3. Si T<0 ir a paso 1, de no ser así ir a paso 4.
4. Si el número se encuentra en minutos, ir a paso 5. De lo contrario ir a paso 6.
5. Realizar una conversión de minutos a horas, H= T/60
6. Si su número presenta horas y minutos ir a paso 7, de lo contrario ir a paso 9.
7. Su número debe de ser multiplicado por el cobro por hora (P=H\*2.5) y el cobro por minuto debe ser multiplicado de la siguiente manera P=m\*0.04167, ir a paso 8
8. Realizar PT=P+Ps, ir a paso10
9. Realizar la operación correspondiente P=H\*2.5, ir a paso 10
10. Se obtuvo la cantidad a pagar por el tiempo en el estacionamiento
11. Fin.

Dados dos números realizar suma, resta, producto y división (considerar el dividendo ≠ 0).

Problema: Realizar la suma, resta, producto y división de dos números.

Datos de entrada: Primer número= A

Segundo número= Z

A/Z

A\*Z

A+Z

A-Z

Restricciones: Z!=0

Datos de salida: Suma, resta, producto y división de dos números

Algoritmo para obtener la suma, resta, producto y división de dos números

1. Inicio
2. Leer A, pasar a paso 3
3. Leer Z, pasar a paso 4
4. Si Z=0, pasar a paso 1, de lo contrario, ir a paso 5.
5. Realizar A+Z, ir a paso 6.
6. Se obtuvo la suma de ambos números, ir a paso 7.
7. Realizar A-Z, ir a paso 8.
8. Se obtuvo la resta de ambos números, ir a paso9.
9. Realizar A\*Z, ir a paso 10.
10. Se obtuvo el producto de ambos números, ir a paso 11.
11. Realizar A/Z, ir a paso 12.
12. Se obtuvo la división de ambos números, ir a paso 13
13. Fin.

Dados tres números; si el primero es negativo, deberá realizar el producto de los tres, y en caso contrario realizará la suma

Problema: Realizar multiplicación o suma de tres números.

Datos de entrada: Primer número= A

Segundo número= Z

Tercer número= M

A\*Z\*M

A+Z+M

Restricciones: *A!=0*

Datos de salida: Suma o producto dependiente del primer número.

Algoritmo para obtener la suma o producto de tres números

1. Inicio
2. Leer A, ir a paso tres
3. Si A es negativa, ir a paso7, de lo contrario ir a paso 4.
4. Si A =0 ir a paso 1, de lo contrario ir a paso 5.
5. Realizar A+Z+M, ir a paso 6
6. Se obtuvo la suma de los números, ir a paso 9.
7. Realizar A\*Z\*M, ir a paso 8.
8. Se obtuvo el producto de los números, ir a paso9.
9. Fin.

**Conclusiones:**

* Elaboración de algoritmos correctos y eficientes en la solución de problemas siguiendo las etapas de Análisis y Diseño pertenecientes al Ciclo de vida del software
* Conocimiento sobre el Ciclo de vida del software.
* Razonamiento y resolución problemas.

***Nota:*** Práctica subida al repositorio, enlace en la bibliografía.

**BIBLIOGRAFIA:**

* *Laboratorio de Computación Salas A y B*. (s. f.). Laboratorio de computaciónFI. Recuperado 08 de Junio de 2021, de <http://lcp02.fi-b.unam.mx/>
* Alonso, A. (2021, 08 Junio). Ale-aloagus/practica1\_fdp. GitHub. <https://github.com/Ale-aloagus/practica1_fdp>