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

Laboratorios de computación

salas A y B

|  |  |
| --- | --- |
| *Profesor:* | KARINA GARCIA MORALES |
| *Asignatura:* | FUNDAMENTOS DE PROGRAMACION |
| *Grupo:* | 19 |
| *No de Práctica(s):* | PRÁCTICA 03: SOLUCIÓN  DE PROBLEMAS Y ALGORITMOS |
| *Integrante(s):* | PARRA PARTIDO BRENDA DANIELA |
| *No. de Equipo de cómputo empleado:* | -- |
| *No. de Lista o Brigada:* | 29 |
| *Semestre:* | 2021-2 |
| *Fecha de entrega:* | MARTES 23 MARZO 2021 |
| *Observaciones:* |  |
|  |  |

CALIFICACIÓN:\_\_\_\_\_\_\_\_\_\_

**Guía práctica de estudio 03: Solución de**

**Problemas y Algoritmos**

* **O B J E T I V O**

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

* **A C T I V I D A D E S**

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

* **C O N C E P T O S**
* **Problema informático**:

Es el conjunto de instancias al cual corresponde un conjunto de soluciones, con una relación que asocia para cada instancia del problema un subconjunto de soluciones (posiblemente vacío).

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". La Ingeniería de Software provee métodos que indican cómo generar software:

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

* **Ciclo de vida del software:**

La ISO (International Organization for Standarization) en su norma 12207 lo define 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.

* **Solución de problemas:**

**El análisis** es el proceso donde se indaga qué es lo que requiere el usuario del sistema de software (análisis de requisitos), nos permite definir las necesidades de forma clara y concisa (especificación de requisitos). Esta etapa es crucial para crear un ***software de calidad***, ya que si no se entiende qué se desea realizar, no se puede generar una solución. Es común caer en ambigüedades debido al mal entendimiento de los requerimientos iniciales. Para ello se deben identificar dos grandes conjuntos dentro del sistema:

* CONJUNTO DE ENTRADA: Está compuesto por todos aquellos datos que pueden alimentar al sistema.
* 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 **el dominio del problema**, es decir, los valores que el problema puede manejar.

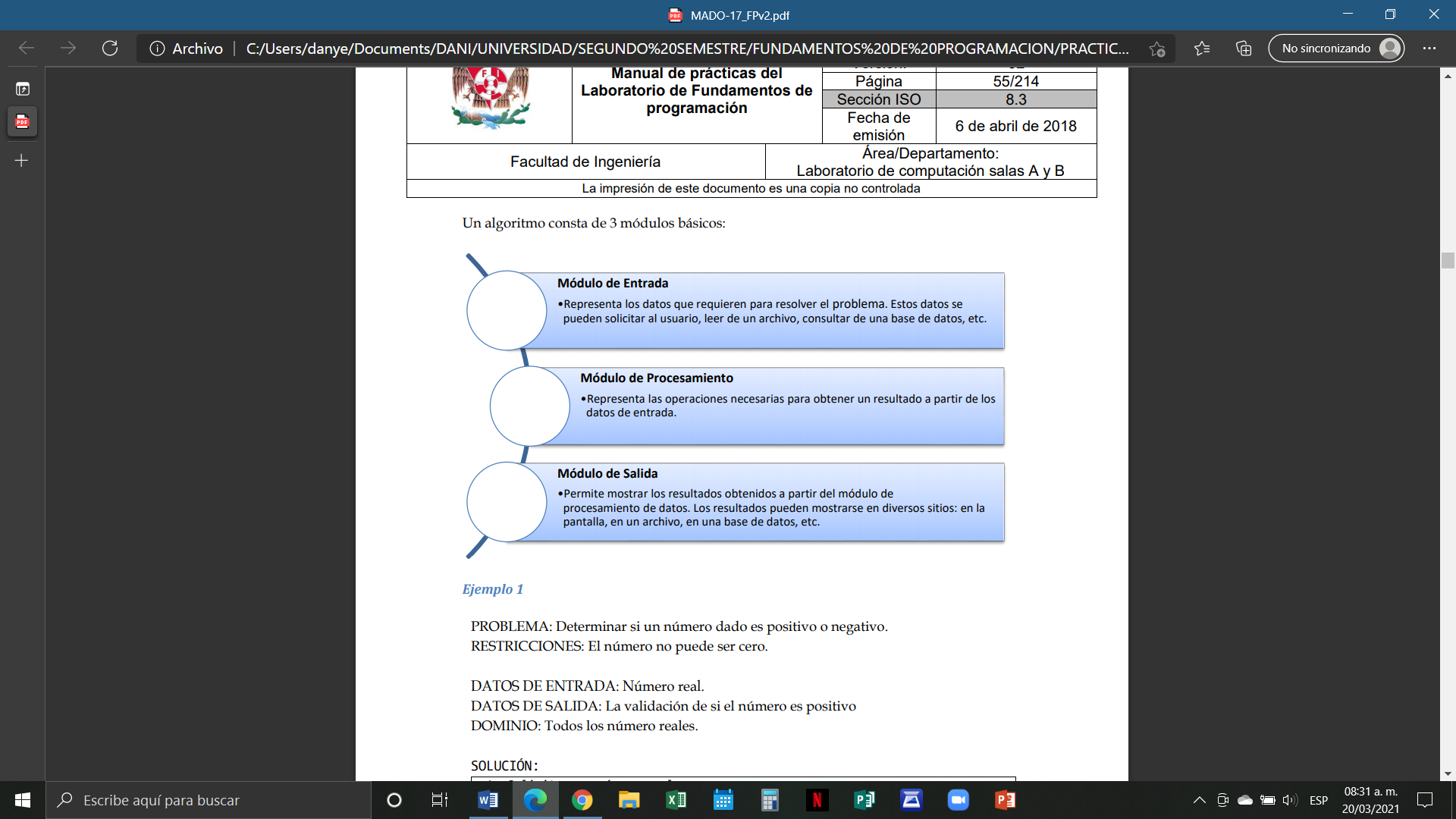
* **ALGORITMOS**

Una vez realizado el análisis, se procede al ***diseño*** de la solución o generación del algoritmo, donde se propone una o varias alternativas y se elige la mejor para iniciar la construcción; un problema matemático es computable si éste puede ser resuelto por un dispositivo computacional. La teoría de la computabilidad estudia los problemas de decisión que pueden ser resueltos con un algoritmo. Un ***algoritmo*** es un conjunto de reglas que pueden ser aplicadas un número ilimitado de veces y están expresadas en un lenguaje específico que realiza un conjunto de pasos/procedimientos/acciones que permiten alcanzar un resultado o resolver un problema. Ellos son es la parte más importante de las ciencias de la computación ya que puede ser creado de manera independiente del lenguaje como de las características físicas del equipo.

\* Características:

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

\*Módulos de un logaritmo:



* **EJEMPLO 1**

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

* **Análisis del problema**
* Datos de Entrada: Número Real, x ∈ R – {0}
* Restricciones: El número no puede ser cero, x ≠ 0
* Datos de Salida: La validación de si el número es positivo
* **Construcción de algoritmo**

1. Inicio, ir al paso 2
2. Leer el valor de x, ir al paso 3
3. Si x ≠ 0 ir al paso 5, en caso contrario si x = 0 ir al paso 4
4. “Error, el valor es incorrecto”, ir al paso 2
5. Si x < 0, ir al paso 6, de caso contrario si x > 0 ir al paso 7
6. “El número es negativo”, ir al paso 8
7. “El número es positivo”, ir al paso 8
8. FIN

* **Prueba de escritorio**

|  |  |  |
| --- | --- | --- |
| **#Intento** | **Valor de x** | **Resultado** |
| **1** | **1** | **Positivo** |
| **2** | **-2** | **Negativo** |
| **3** | **0** | **--** |

* **EJEMPLO 2**

***PROBLEMA: Obtener el mayor de dos números dados***

* **Análisis del problema**
* Datos de Entrada: Número real, x , y ∈ R
* Restricciones: Los números de entrada deben ser diferentes. X ≠ Y
* Datos de Salida: La impresión del número más grande
* **Construcción de algoritmo**

1. Inicio, ir al paso 2
2. Leer el valor de X, Y, si X = Y, ir al paso 3, si X≠Y ir al paso 4
3. “Error, datos invalidos”, ir al paso 2
4. Si X > Y , ir al paso 5, en caso contrario si X < Y ir al paso 6
5. X “es mayor que” Y , ir al paso 7
6. Y “es mayor que” X, ir al paso 7
7. Fin

* **Prueba de escritorio**

|  |  |  |  |
| --- | --- | --- | --- |
| **#Intento** | **Valor de X** | **Valor de Y** | **Resultado** |
| **1** | **1** | **-1** | **1 mayor que -1** |
| **2** | **-5** | **5** | **5 mayor que -5** |
| **3** | **8** | **8** | **--** |

* **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.

* **Análisis del problema**
* Datos de Entrada: Número entero positivo, n > 0
* Restricciones: El número de entrada debe ser entero y no puede ser negativo, n ∈ N – { 0 }, a = 1, n = a + m
* Datos de Salida: La impresión de la factorial del número deseado.
* **Construcción de algoritmo**

1. Inicio, ir al paso 2
2. Leer el valor de n, ir al paso 3
3. Si n = 0, ir al paso 4, si n < 0 ir al paso 5 en caso contrario si n > 0 ir al paso 6
4. “El factorial de 0 es 1”, ir al paso 9
5. “Error, dato inválido”, ir al paso 2
6. Cuando n > 0, el factorial es igual a n! = (a)\*(a+1)\*(a+2)\*(a+3)\*...\*(a+m), ir al paso 7
7. Realizar la operación de n!, ir al paso 8
8. n! “Es el factorial de” n, ir al paso 9
9. Fin

* **Prueba de escritorio**

|  |  |  |  |
| --- | --- | --- | --- |
| **#Intento** | **n** | **Operación de n**! | **Resultado de n**! |
| **1** | **0** | **-** | **1** |
| **2** | **-3** | **--** | **--** |
| **3** | **10** | **(1)\* (1+1)\* (1+2)\* (1+3)\* (1+4)\* (1+5)\* (1+6)\* (1+7)\* (1+8)\* (1+9)\*** | **3,628,800** |

* **T A R E A - 1 – Ejercicio 1**
* PROBLEMA: Seguir el algoritmo para obtener una figura
* ENTRADA: Hoja tamaño carta en limpio, regla y lápiz.
* SALIDA: **Estrella de cinco puntas**

* **T A R E A – 1 – Ejercicio 1**
* PROBLEMA: Seguir el algoritmo para obtener una figura
* ENTRADA: Hoja tamaño carta en limpio, regla y lápiz.
* SALIDA: **Estrella de 6 puntas**
* **T A R E A - 1 –** Comparación

|  |  |
| --- | --- |
| **EJERCICIO 1** | **EJERCICIO 2** |
| **En ambos logaritmos existe una ambigüedad, ya que no especifican a que paso debe ir el usuario al concretar cada uno de los puntos, no están expresando lo “obvio” y dejando de ser preciso.** | |
| **El problema me parece mal planteado, ya que el logaritmo no debería ser el problema, sino la forma en resolver el verdadero problema, que sería construcción de una figura geométrica.** | |
| **En ninguno de ambos ejercicios colocan restricciones en el análisis de datos cuando es algo fundamental que no debe pasarse por alto, por lo que ocurre en una segunda ambigüedad.** | |
| **Para todo diseño de algoritmo se deben realizar y culminar tres fases, (análisis, construcción y verificación) por lo que no están cumpliendo con la tercera fase, requiere forzosamente realizar pruebas de escritorio para asegurar que funciona correctamente el logaritmo diseñado.** | |

* **T A R E A – 2**

**Dar solución a un**

**PROBLEMA**

Para poder dar solución a un problema es fundamental entender y comprender en su totalidad la necesidad y requerimientos que solicita nuestro usuario, por lo que debemos ser minuciosos al comenzar a diseñar nuestro algoritmo que dará solución al problema. Por el contrario al no tener en claro lo que buscamos obtener, se caerá en ambigüedades y es imposible generar una solución clara y sencilla.





Por lo tanto primero debemos identificar con que datos se cuenta, los cuales serán nuestros datos de entrada, así como el resultado al que se desea llegar, y por último se deben añadir las restricciones que nos permitirá el desarrollo de un algoritmo eficaz, es decir, que nos producirá el resultado esperado.

* **T A R E A – 3**

***Ciclo de vida del SOFTWARE***

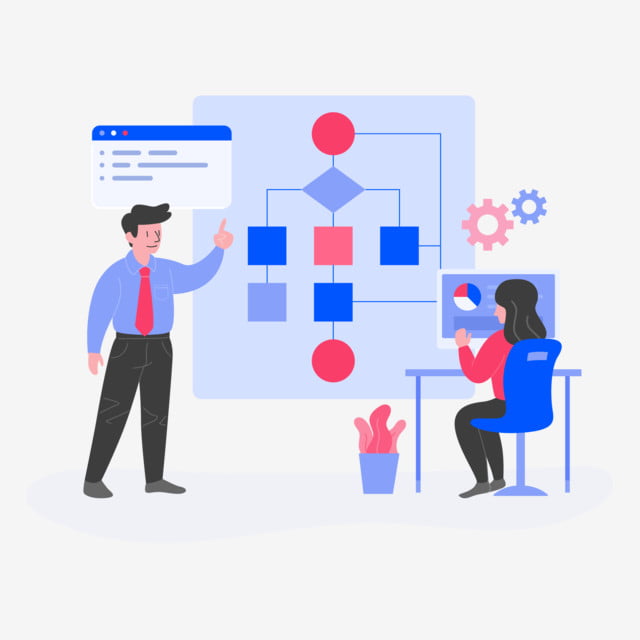
El ciclo de vida del software es la estructura que contiene los procesos, actividades y tareas relacionadas con el desarrollo y mantenimiento de un producto de software, abarcando la vida completa del sistema, desde la definición de los requisitos hasta la finalización de su uso y este consta de siete fases:



1. **Definición de necesidades**

En esta el usuario especificará que es lo que requiere que nosotros desarrollemos para dar solución a su problema o necesidad.

1. **Análisis**



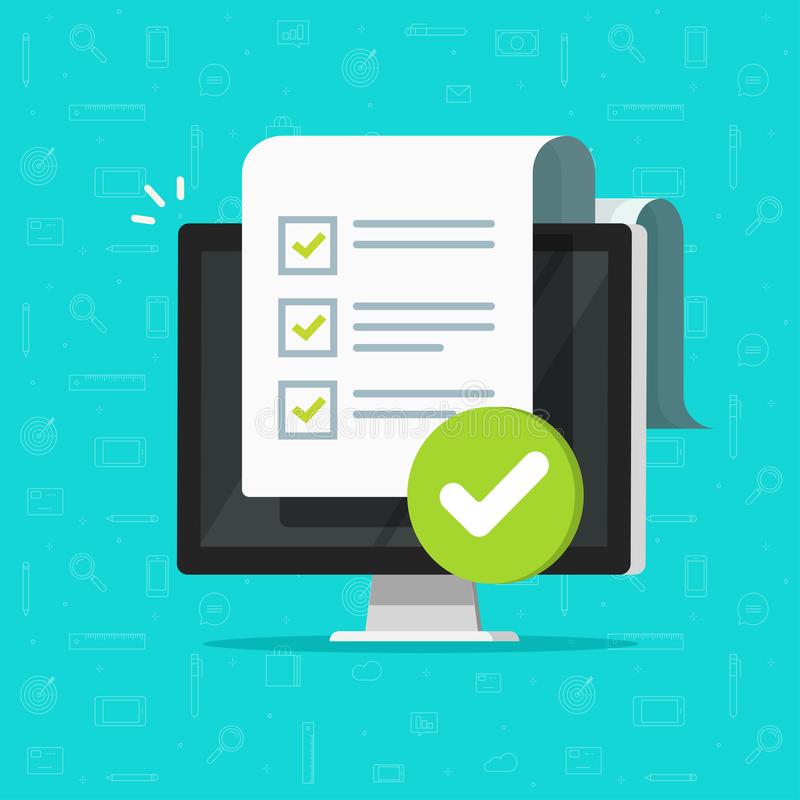
En ella una vez comprendido el problema y detectado cual es la necesidad del usuario, se procede a seleccionar con que datos contamos (Datos de entrada), que cosas no podemos realizar o debemos cumplir (restricciones) y que resultado deseamos obtener (Datos de salida).

1. **Diseño**



En ella como su nombre lo indica es donde se plantean una serie de pasos a seguir, con un inicio, un fin y especificaciones que nos permitirán de forma eficaz y sencilla llegar a nuestro resultado.

1. **Pruebas**



En esta etapa se realizará la comprobación de diseño que se hizo en la etapa pasada, para verificar el correcto funcionamiento del logaritmo y comprobar que no existe errores o ambigüedades.

1. **Validación**



En ella el usuario comprobará que el algoritmo desarrollado cumple con las condiciones y características que estableció en un inicio, de ser así, se genera la validación del mismo.

1. **Mantenimiento y evolución**



En esta etapa el programador del algoritmo se asegurará de que siga funcionando correctamente, así como implementar mejoras o actualizaciones para potencializar su funcionamiento en beneficio del usuario.

* **T A R E A – 4**

***PROBLEMA: En una playa el estacionamiento cobra $ 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.***

* **Análisis del problema**
* Datos de Entrada: Estacionamiento $2.5 por hora
* Restricciones: x 1 hora =$ 2.5 , x = horas, x ≥ 0 , x \* 2.5 = y , las horas serán la parte entera y los minutos como la parte decimal
* Datos de Salida: Monto a pagar por el servicio de estacionamiento
* **Construcción de algoritmo**

1. Inicio, ir al paso 2
2. Leer el valor de x, ir al paso 3
3. Si x < 0, ir paso 4, si x = 0 ir al paso 5, en caso contrario si x > 0, ir al paso 6
4. “Error, valor invalido”, ir al paso 2
5. “Usted debe pagar el monto de $ 0”, ir al paso 8
6. Efectuar la operación de x \* 2.5, ir al paso 7
7. “Usted debe pagar el monto de” y, ir al paso 8
8. Fin

* **Prueba de escritorio**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **#Intento** | **Tiempo** | **Valor de X** | **Valor de Y** | **Resultado** |
| **1** | **-2** | **--** | **--** | **--** |
| **2** | **0:00** | **0** | **0** | **$0** |
| **3** | **3:40** | **3.40** | **8.5** | **$8.5** |

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

* **Análisis del problema**
* Datos de Entrada: Dos números x, y
* Restricciones: x , y ∈ Z – { 0 }
* Datos de Salida: Resultado de la operación + , - , \* , / de dos números
* **Construcción de algoritmo**

1. Inicio, ir al paso 2
2. Leer el valor de x , y, ir al paso 3
3. Cuando x = y , ir al paso 4, si y=0, ir al paso 10 en caso contrario si x ≠ y ir al paso 14
4. “La resta de” x “menos” y “es igual a cero”, ir al paso 5
5. Efectuar la operación x + y, ir al paso 6
6. “La suma de” x “más” y “es igual a“ x+y, ir al paso 7
7. “La división de” x “entre” y “es igual a 1”, ir al paso 8
8. Efectuar la operación multiplicación de x\*y, ir al paso 9
9. “La multiplicación de” x “por” y “es igual a” x\*y, ir al paso 22
10. “La suma de” x “más ” y “es igual a ”x, ir al paso 11
11. “La resta de” x “menos” y “es igual a” x, ir al paso 12
12. “La multiplicación de” x “por” y “es igual a 0”, ir al paso 13
13. “La división de” x “entre” y “no existe”, ir al paso 22
14. Efectuar la operación suma de x+y, ir al paso 15
15. “La suma de” x “más ” y “es igual a ” x+y, ir al paso 16
16. Realizar la operación de x-y, ir al paso 17
17. “La resta de” x “menos” y “es igual a” x-y, ir al paso 18
18. Realizar la operación x\*y, ir al paso 19
19. “La multiplicación de” x “por” y “es igual a” x\*y, ir al paso 20
20. Realizar la operación de x/y, ir al paso 21
21. “La división de” x “entre” y “es igual a” x/y, ir al paso 22
22. Fin

* **Prueba de escritorio**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **#Intento** | **Valor de X** | **Valor de Y** | **Resultado**  **Suma** | **Resultado**  **Resta** | **Resultado**  **Multiplicación** | **Resultado**  **División** |
| **1** | **2** | **2** | **4** | **0** | **4** | **1** |
| **2** | **5** | **0** | **5** | **5** | **0** | **--** |
| **3** | **4** | **6** | **10** | **-2** | **24** | **2/3** |

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

* **Análisis del problema**
* Datos de Entrada: x, y, z ∈ R
* Restricciones: si x ≥ 0, será multiplicación, si x < 0 será suma
* Datos de Salida: Resultado de la suma o producto de tres números.
* **Construcción de algoritmo**

1. Inicio, ir al paso 2
2. Leer el valor de x, y, z, ir al paso 3
3. Si x < 0 ir al paso 4, de lo contrario si x > 0 ir al paso 6
4. Realizar la operación multiplicación de x \* y \* z , ir al paso 5
5. “El resultado de la operación multiplicación es” x \* y \* z , ir al paso 8
6. Realizar la operación suma de x + y + z, ir al paso 7
7. “El resultado de la operación suma es” x + y + z, ir al paso 8
8. Fin

* **Prueba de escritorio**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **#Intento** | **Valor de X** | **Valor de Y** | **Valor de z** | **Operación** | **Resultado** |
| **1** | **0** | **0.5** | **9.1** | **0\*0.5\*9.1** | **0** |
| **2** | **2** | **4** | **-5** | **2\*4\*-5** | **-40** |
| **3** | **-1** | **9** | **1** | **-1+9+1** | **9** |

**C O N C L U S I O N E S**

Al término de la realización de esta práctica se logró cumplir con el objetivo esperado, planteado al inicio de la misma, el cual fue la elaboración de algoritmos que cumplieran con ser correctos y eficientes, una actividad que fue bastante esclarecedora ya que se logró concretar los conocimientos vistos en clase así como aclarar cualquier duda que aún existiera. Con el desarrollo de esta práctica nos hizo ver claramente un punto vital y fundamental al momento de diseñar un algoritmo, el cual es el identificar correctamente cual es el problema a resolver, que condiciones o restricciones pide mi usuario así como el resultado al que se desea llegar, ya que sin una buena definición del problema y del análisis, podremos estar expuestos a cometer errores, como son caer en ambigüedades o bien, que nuestro algoritmo no realice la función que nuestro usuario necesita que haga, por lo tanto, es de suma importancia comenzar minuciosamente desde el principio del diseño del algoritmo para que todo lo demás sea igualmente correcto. Por otro lado se vio el concepto del Ciclo de vida del Software, el cual consta de siete etapas y de las cuales en su mayoría ya las habíamos estudiado en clase. Y como comentario final me pareció una práctica bastante enriquecedora ya que a partir de ella logramos ver nuestro avance en el curso al igual como nuestras oportunidades para mejorar y en qué puntos necesitamos reforzar.

**B I B L I O G R A F Í A**

* Facultad de Ingeniería. (2015). *Laboratorio de Computación Salas A y B*. Recuperado de: <http://lcp02.fi-b.unam.mx/>
* Solano Gálvez. (2018). *Manual de prácticas del laboratorio de Fundamentos de programación*. Recuperado de: <http://lcp02.fi-b.unam.mx/>