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

|  |  |
| --- | --- |
| *Profesor:* | Ing. Karina Garcia Morales |
| *Asignatura:* | Fundamentos de programacion |
| *Grupo:* | 20 |
| *No de Práctica(s):* | 03 |
| *Integrante(s):* | Juan Pablo Rios Alvarez |
| *No. de Equipo de cómputo empleado:* | N/A |
| *No. de Lista o Brigada:* | 41 |
| *Semestre:* | 2021-1 |
| *Fecha de entrega:* | 29 de octubre de 2020 |
| *Observaciones:* |  |
|  |  |

**Solución de Problemas y Algoritmos.**

**Objetivo:**

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

**Desarrollo de la práctica:**

En esta práctica se comenzó por leer el objetivo y las actividades que realizaríamos, uno de mis compañeros leyó la introducción en la cual observamos las actividades que se realizaron, las cuales fueron: 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.

**Introducción:**

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"

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

* Planeación y estimación del proyecto: En esta parte se ven los tiempos, los alcances, lo que se desea hacer, repartiendo por tiempos la estimación de cada paso de la solución del problema.
* Análisis de requerimientos del sistema y software: Que software será el más útil para la tarea, porque usará un módulo de registro, de login y se tiene que ver cuál es el más conveniente.
* Diseño de la estructura de datos, la arquitectura del programa y el procedimiento algorítmico: Diseño del algoritmo, diagrama de flujo, y el pseudocódigo con todas las pruebas que realizamos.
* Codificación: Ejecución
* Pruebas y mantenimiento (validación y verificación): Aprobar que lo que se nos solicita se cumpla.

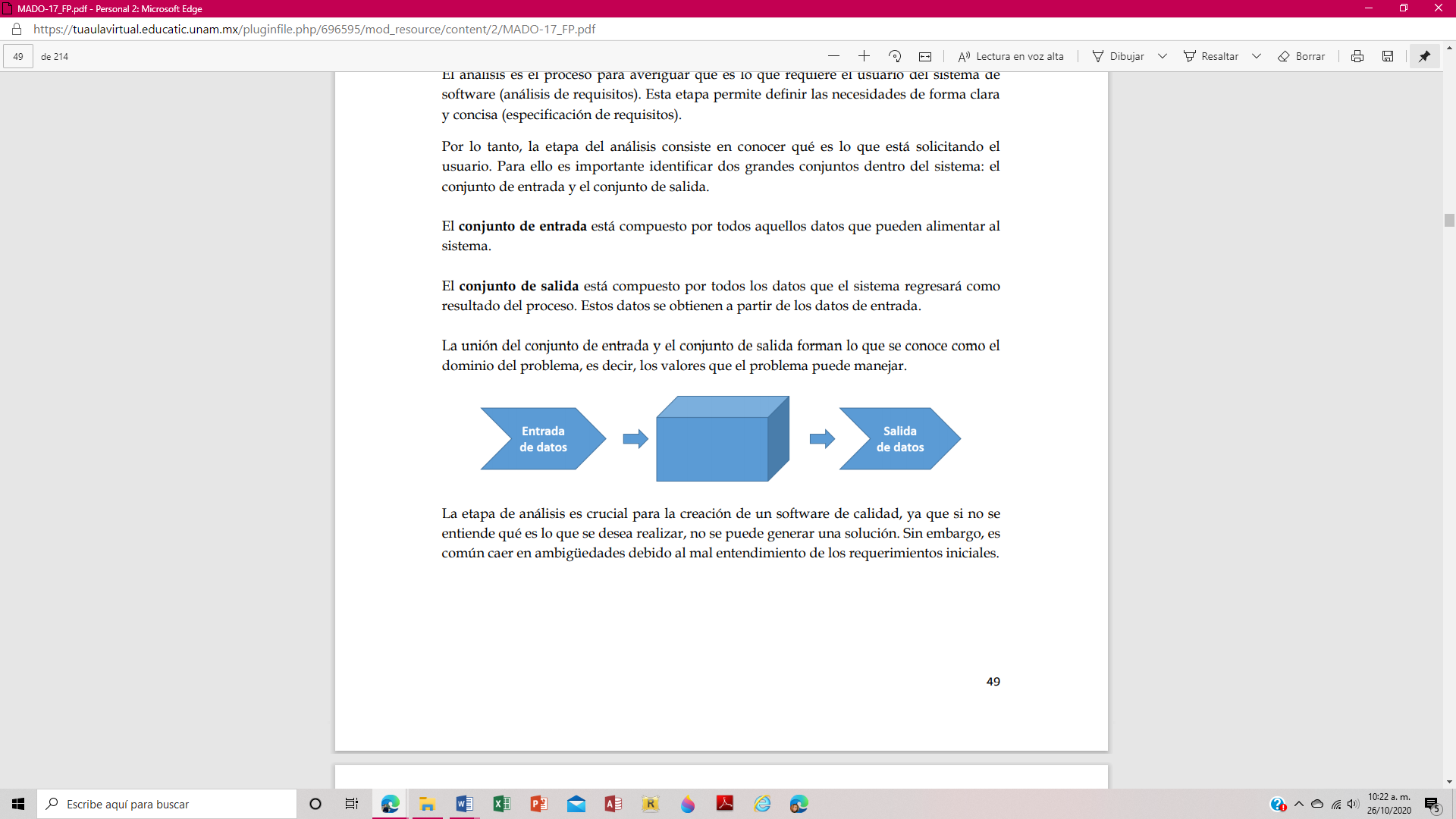
*Ciclo de vida del software*

La ISO la define como un marco de referencia de lo que se debe hacer en desarrollo, la explotación y el mantenimiento de un producto de software, desde la definición hasta la finalización de su uso.

*Solución de problemas*

El análisis busca saber que quiere el usuario para definir sus necesidades de forma clara y concisa. Sus 2 conjuntos son el conjunto de entrada que son los datos que alimentan al sistema y el conjunto de salida que son los datos que regresa el sistema con base en 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.



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.

Ejemplo 1

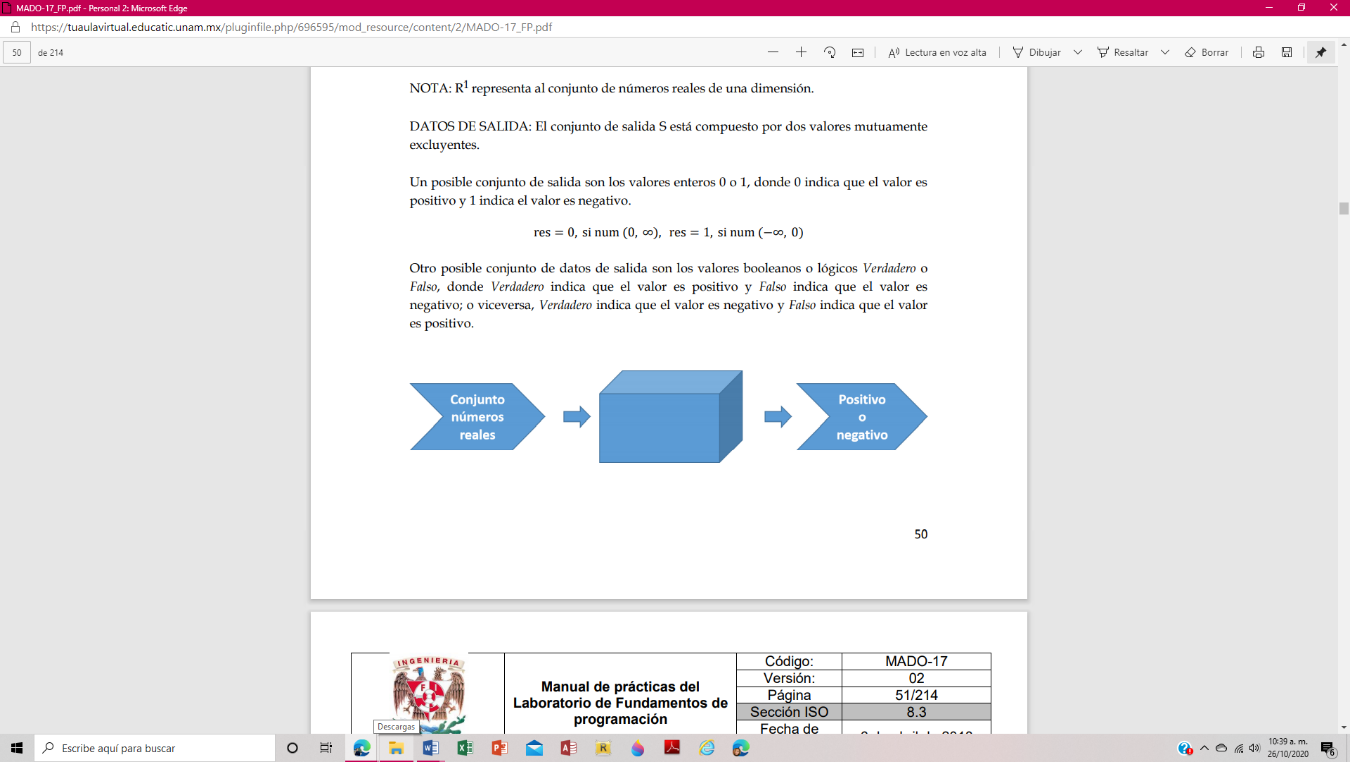
Problema 1

Se trata de saber si un número es positivo o negativo. En la siguiente condición nos dice que puede introducirse cualquier número real, excepto 0.

E ⊂ R1, donde núm. ∈ E de (−∞, ∞) − {0}

Esta condición nos dice que aparecerá como resultado 0 si el número es positivo y 1 si el resultado es negativo.

res = 0, si núm. (0, ∞), res = 1, si núm. (−∞, 0)



Ejemplo 2

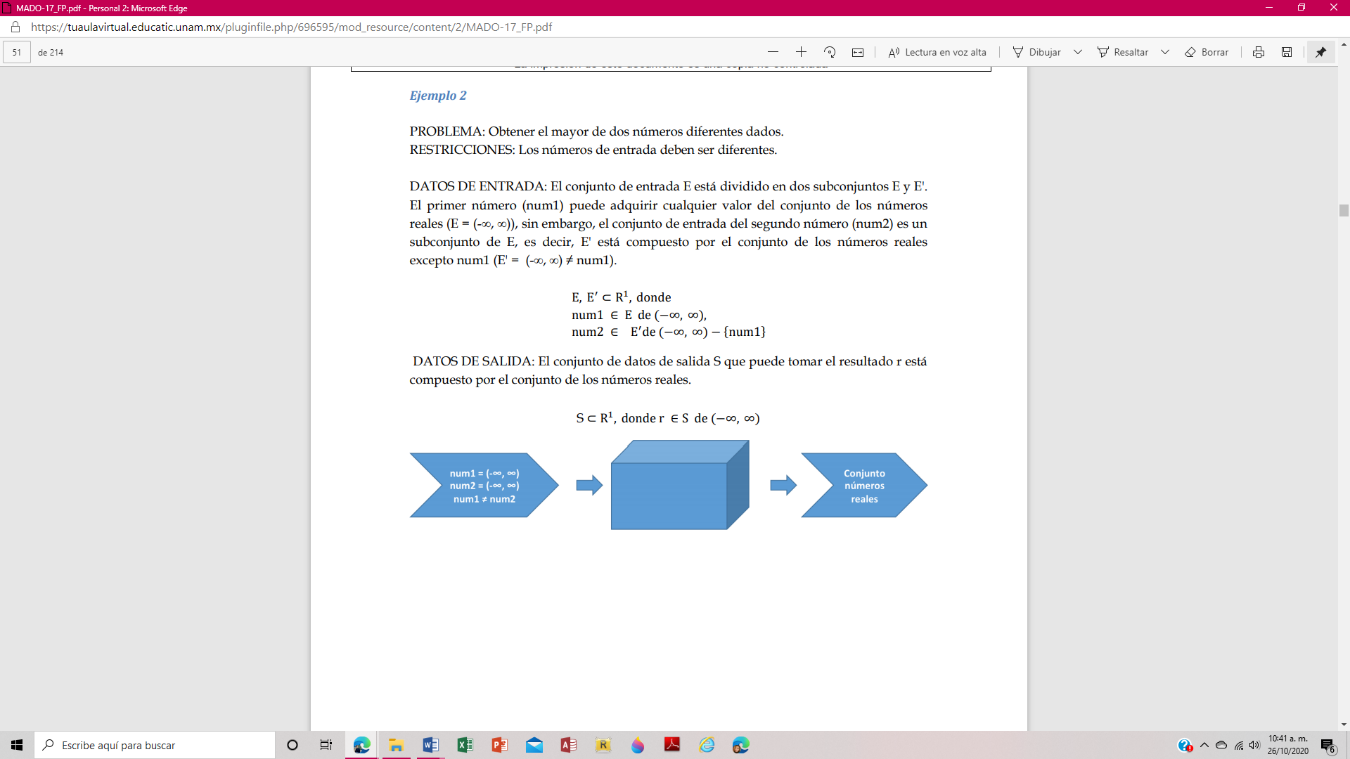
Se trata de seleccionar cuál de los 2 números que vayamos a introducir es el más grande.

Esta condición nos dice que podemos insertar cualquiera de los números reales excepto 2 iguales

E, E′ ⊂ R1, donde num1 ∈ E de (−∞, ∞), num2 ∈ E′ de (−∞, ∞) − {num1}

Esta condición nos dice que los datos de salida abarcan todos los reales

S ⊂ R1, donde r ∈ S de (−∞, ∞)



Ejemplo 3

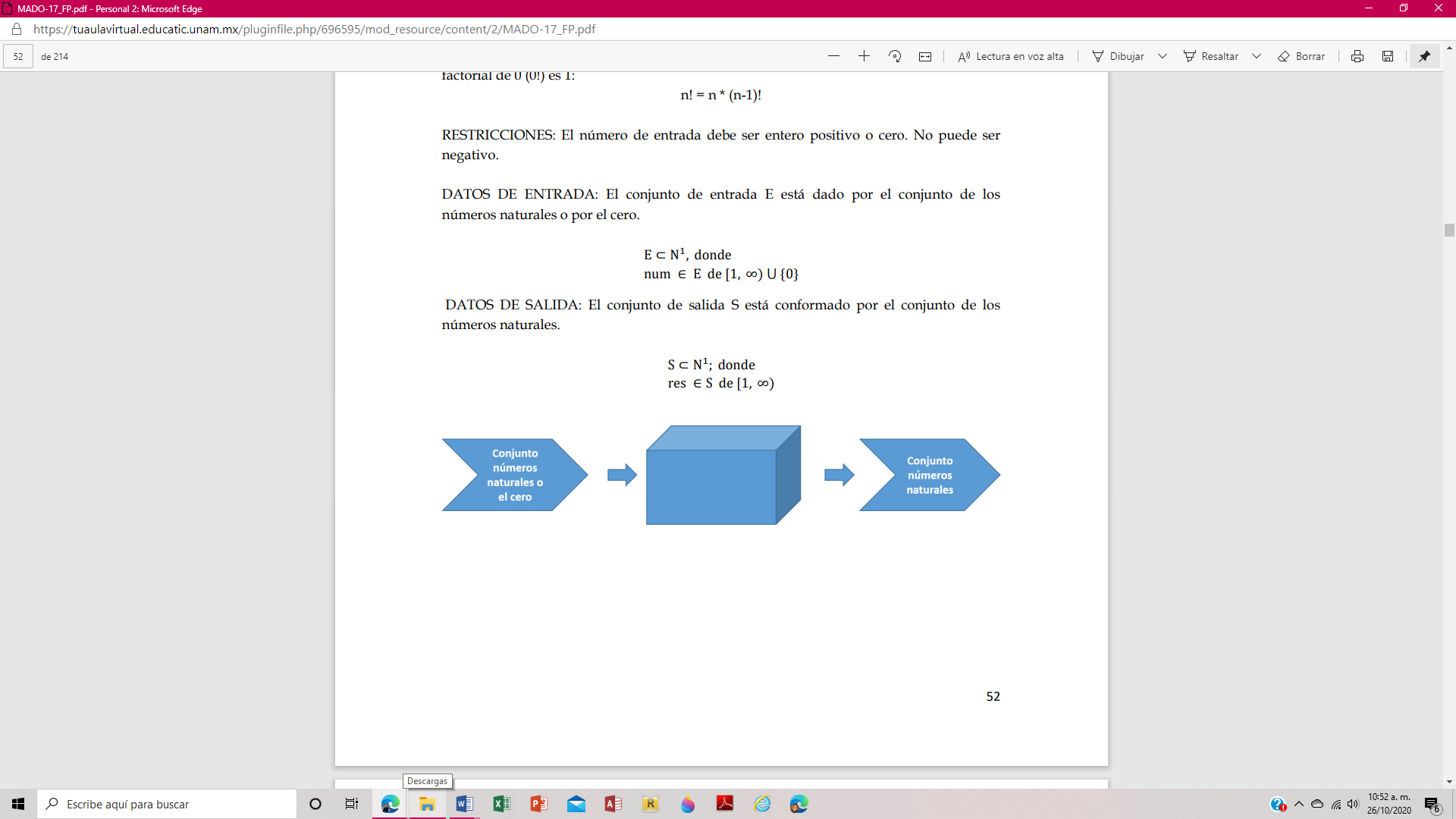
DE: Se trata de obtener la factorial de un número cuya fórmula es: n! = n \* (n-1)!

R: Esta condición nos dice que N1 puede ser cualquier número, excepto uno negativo

E ⊂ N1, donde núm. ∈ E de [1, ∞) ⋃ {0}

DS: Esta condición nos dice que los datos de salida deben ser positivos

S ⊂ N1; donde res ∈ S de [1, ∞)



Algoritmos

Dentro del ciclo de vida del software, la creación de un algoritmo se encuentra en la etapa de diseño, o sea entre la entrada y la salida. Es un conjunto de reglas, expresadas en un lenguaje específico, para realizar una tarea y pueden ser aplicadas infinitas veces. Un algoritmo es la parte más importante y durable de las ciencias de la computación porque puede crearse independientemente tanto del lenguaje como de las características físicas del equipo que lo va a ejecutar.

Debe tener estas características:

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

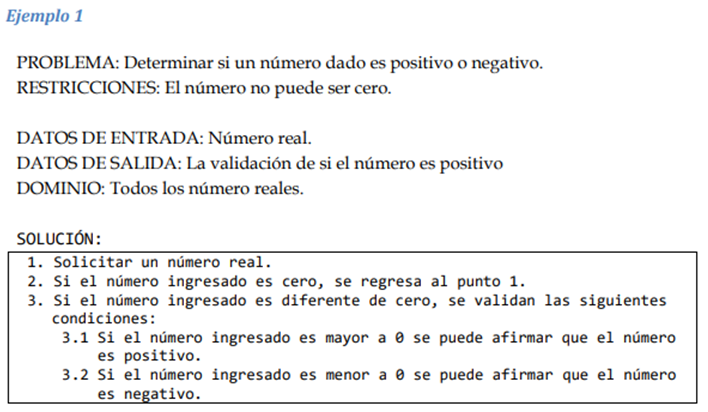
Eficaz: Que produzca el efecto esperado

Para hacer un algoritmo se deben seguir estos pasos:

ANÁLISIS DEL PROBLEMA: Conocer los datos de entrada, restricciones y los datos de salida

CONSTRUCCION DEL ALGORITMO: Describir detalladamente los pasos

VERIFICACIÓN DEL ALGORITMO: Prueba de escritorio

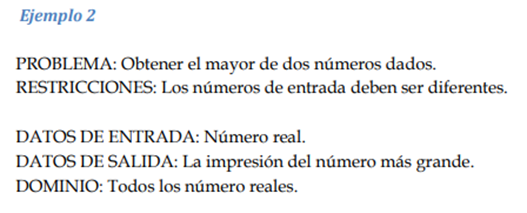


Mi algoritmo para este problema es:

1. Inicio.
2. Colocar un número real.
3. Si el número ingresado es cero, volver al paso 2, de lo contrario ir al paso 4.
4. Si el número es mayor que 0 ir al paso 5, si es menor que 0 ir al paso 6.
5. El número es positivo, ir al paso 7.
6. El número es negativo, ir al paso 7.

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)



Mi solución:

1. Inicio
2. Solicitar 2 números diferentes e ir al paso 3
3. Si es igual a 0 alguno de esos números regresar al paso 2, si es diferente de 0 continuar.
4. Si ambos son diferentes, se puede cumplir lo siguiente:

4.1-Si el primer número real es mayor al segundo ir al paso 5.

4.2-Si el segundo número real es mayor ir al paso 6.

5. El primer número es el más grande de los dos

6. El segundo número es el mayor de los números

7-Fin

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.

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

Datos de entrada: Número entero.

Datos de salida: La impresión de la factorial del número.

Dominio: Todos los números naturales positivos.

Mi solución:

1-Inicio

2-Solicitar un número entero.

3-Si el número entero es menor a cero regresar al punto 1.

4-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.

5-Si la variable contador es menor o igual al número entero de entrada se realiza lo siguiente:

-Se multiplica el valor del variable contador con el valor de la variable factorial. El resultado se almacena en la variable factorial.

-Se incrementa en uno el valor del variable contador.

-Regresar al punto 5.

6. Si la variable contador no es menor o igual al número entero se muestra el resultado almacenado en la variable factorial.

7-Fin

**EJERCICIOS DE TAREA**

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

Diseñando un plan y ejecutarlo, los pasos para hacerlo son:

1. análisis previo del problema

## 2. diseño del algoritmo

## 3. configuración del entorno de desarrollo y codificación

## 4. compilación y pruebas

## 5. documentación y mantenimiento

## 2. Describe las fases del ciclo de vida del desarrollo de software e ilústralas

## 

3. 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.

Análisis.

Calcular el monto a pagar del uso del estacionamiento por determinado tiempo

Datos de Entrada: $2.5 por hora, He-Hora de entrada, Hs-Hora de salida, Me-Minuto de entrada, Ms-Minuto de salida.

Restricción: Horas y minutos, números positivos

Datos de Salida: Monto a pagar.

Algoritmo:

1. Inicio

2. Introducir los valores correspondientes al tiempo que se usó el estacionamiento, siendo números positivos e ir al paso 3.

- Leer He

- Leer Hs

- Leer Me

- Leer Ms

3. Sustituir valores en la fórmula: (Hs-He)\*60 + (Ms-Me) para conocer los minutos totales e ir al paso 4.

4. Dividir el total de min/60, para conocer las horas a pagar e ir al paso 5.

5. Para saber el monto a pagar se aplica la fórmula: horas a pagar\*2.5 e ir al paso 6.

6. Fin

Análisis

Realizar las diferentes operaciones matemáticas con dos números

Datos de Entrada: dos números representado por a y b.

Restricción: El dividendo debe ser ≠ 0, números enteros.

Datos de Salida: Tipo de Operación y resultado.

Algoritmo:

1. Inicio
2. Declarar variables a para el primer número, b para el segundo, s para suma, r para resta, m de multiplicación y d para división e ir al paso 3.
3. Introducir valores enteros para ambos números
4. Si algún número es negativo, o no pertenece a los números enteros regresar al paso 3, en caso contrario ir al paso 5.

5. Realizar operación de suma a+b=s e ir al paso 6.

-Realizar operación de resta a-b=r e ir al paso 6

-Realizar operación de multiplicación a\*b=m e ir al paso 6

-Realizar operación de división a/b=d, sólo si el dividendo>0 en caso contrario regresar al paso 3, si no es así al 6.

6. Resultado de mi operación.

7. Fin

Análisis

Datos de entrada: 3 números

Restricción: números enteros, sólo suma o división, signo positivo o negativo.

Datos de salida: Operación entre los 3 números.

Algoritmo

1. Inicio
2. Introducir 3 números enteros, a, b y c, e ir al paso 3.
3. Si a es un número negativo ir al paso 4, en caso contrario ir al paso 5,
4. Realizar la operación (-a) (b) (c) e ir al paso 6
5. Realizar la operación a + b + c e ir al paso 6
6. Fin

Conclusión

En esta práctica hemos visto la resolución de problemas, desde que identificamos nuestra necesidad, comenzando a separar nuestros datos para una mejor comprensión, además algoritmos permiten encontrar la solución a problemas computables. Intuitivamente las personas efectuamos cotidianamente una serie de pasos, procedimientos o acciones que nos permitan alcanzar algún resultado o resolver un problema

Referencias

* Facultad de Ingeniería. (2015).Manual de Prácticas de Laboratorio Salas A y B. Consultado el 27 de Octubre de 2020 en: <http://lcp02.fi-b.unam.mx/>