Problemas y Algoritmos

Nombre: **Jhoan Raul Acosta Piedrahita**

Código: **20241135035**

Docente: **Karen Patricia Gaitan de los Rios**

Tema: **Construyendo algoritmos sólidos: probando, analizando y comentando**

Fecha: 17 de septiembre de 2025

Índice general

Capítulo 1

Soluciones de problemas

El análisis es el proceso para averiguar que es lo que requiere el usuario del sistema de software en los análisis requeridos. La etapa del análisis consiste en conocer que es lo que esta solicitando el usuario, se definen dos grandes conjuntos:

- **Conjunto de entrada:** Está compuesto por todos aquellos datos que alimentan al sistema.
- **Conjunto de salida:** Está compuesto por todos los datos donde se da un resultado del proceso.

La unión de estos dos conjuntos forma lo que se conoce como el dominio del problema, los valores que maneja el problema.

Capítulo 2

Algoritmos

Dentro del ciclo de vida del software, se crea un algoritmo en su etapa de diseño, durante este diseño se busca proponer una o varias alternativas viables para dar solución al problema. Un problema matemático es computable si este puede ser resuelto. Existe una teoría de la computabilidad que es la parte de la computación que estudia los problemas de decisión que son resueltos en algoritmos.

Un algoritmo se define como un conjunto de reglas, que permiten alcanzar un resultado o resolver un problema. Estas reglas pueden ser aplicadas un numero de veces ilimitadas sobre una situación particular. Es la parte mas importante y durable de las ciencias de la computación ya que puede ser creado de manera independiente. Las principales características son:

- Preciso
- Definido
- Finito
- Correcto
- Debe tener una salida y debe ser perceptible
- Sencillo y legible
- Eficiente
- Eficaz

2.1. Forma correcta para solucionar un algoritmo

Resultados del análisis del problema: Con que datos se cuenta, cuales son necesarios como valores de entrada, que restricciones deben considerarse.

Construcción del algoritmo: Se refiere a la descripción detallada de los pasos que deben segurise para resolver el problema.

Verificación del algoritmo: Consiste en el seguimiento del mismo, empleando datos que son representativos del problema se desea resolver.

2.2. Módulos Básicos

Módulo de entrada
Módulo de procesamiento
Módulo de salida

2.3. Variables

Un algoritmo requiere del uso de variables porque guardan el valor (numérico o no numérico) de los datos de entrada, pueden ser utilizados para almacenar datos generados en el proceso y datos de salida.

Capítulo 3

Ejemplo 1: Número positivo o negativo

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 indicación de si el número es positivo o negativo.
DOMINIO: Todos los números reales.

3.1. Solución

1. Solicitar un número real y almacenarlo en una variable
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:
   - Si el número ingresado es mayor a 0 se puede afirmar que el número es positivo.
   - Si el número ingresado es menor a 0 se puede afirmar que el número es negativo.

3.2. Prueba de escritorio

Iteración	X	Salida
1	5	El número es positivo
1	-29	El número es negativo
1	0	-
2	0	-
3	0	-
4	100	El número es positivo

Estas tablas muestran que el algoritmo responde bien en tres situaciones diferentes:
Cuando el número es positivo.
Cuando el número es negativo.
Cuando el número es cero.
Gracias a esta simulación en papel, se valida que el algoritmo cumple con lo esperado antes de llevarlo a código.

Capítulo 4

Ejemplo 2: Mayor de dos números

PROBLEMA: Obtener el mayor de dos números dados.
RESTRICCIONES: Los números de entrada deben ser diferentes.
DATOS DE ENTRADA: Dos números reales.
DATOS DE SALIDA: La escritura del número más grande.
DOMINIO: Todos los números reales.

4.1. Solución

1. Solicitar un primer número real y almacenarlo en una variable.
2. Solicitar un segundo número real y almacenarlo en otra variable.
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:
   - 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.
   - 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.

4.2. Prueba de escritorio

Iteración	X	Y	Salida
1	5	6	El segundo numero es el mayor de los números
1	-99	-222.2	El primer número es el mayor de los números
1	15	15	-
2	15	15	-
3	15	15	-
4	15	10	El primer número es el mayor de los números

Con estas pruebas se valida que el algoritmo funciona correctamente en todos los casos:
Cuando el segundo número es mayor.
Cuando el primer número es mayor.
Cuando los números son iguales (y se vuelve a pedir que ingresen otros valores).

Capítulo 5

Ejemplo 3: Factorial

PROBLEMA: Obtener el factorial de un número dado.
RESTRICCIONES: El número de entrada debe ser entero y no puede ser negativo.
Nota: 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.
DATOS DE ENTRADA: Número entero.
DATOS DE SALIDA: El factorial del número.
DOMINIO: Todos los números naturales y el cero.

5.1. Solución

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

5.2. Prueba de escritorio

Iteración	X	Factorial	Contador	Salida
1	0	1	2	El factorial es 1

1	-2	1	2	-
2	-67	1	2	-
3	5	1	2	-
4	5	2	3	-
5	5	6	4	-
6	5	24	5	-
7	5	120	6	El factorial es 120
1	7	1	2	-
2	7	2	3	-
3	7	6	4	-
4	7	24	5	-
5	7	120	6	-
6	7	720	7	-
7	7	5040	8	El factorial es 5040

La prueba de escritorio demuestra que el algoritmo:

Rechaza números negativos.

Calcula correctamente el factorial de 0 (caso especial).

Realiza la multiplicación paso a paso hasta obtener el factorial de cualquier número entero positivo.

Capítulo 6

Ejemplos interesantes

6.1. Ejemplo 1

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:

$$n! = n \times (n-1)!$$

RESTRICCIONES: El número de entrada debe ser entero positivo o cero. No puede ser negativo.

DATOS DE ENTRADA: El conjunto de entrada E está dado por el conjunto de los números naturales o por el cero.

$$E \subset N1, \quad num \in E \; de \; [1, \infty) \cup \{0\}$$

DATOS DE SALIDA: El conjunto de salida S está conformado por el conjunto de los números naturales.

$$S \subset N1; \quad res \in S \; de \; [1, \infty)$$

El algoritmo para calcular el factorial es simple, intuitivo y eficiente para valores pequeños o medianos de n. Sin embargo, para valores muy grandes se deben considerar optimizaciones en el manejo de grandes enteros y técnicas más avanzadas para evitar problemas de tiempo y memoria. En contextos prácticos como combinatoria, estadística o probabilidad, muchas veces basta con trabajar con logaritmos o aproximaciones.

El algoritmo trabaja con números enteros y utiliza la operación modulo % como criterio para clasificar.

El conjunto de salida es mutuamente excluyente.

Aprendizaje

Definir claramente del dominio de entrada, los números enteros y el rango de salida los números par e impar.

Reconocer la operación modulo como una herramienta clave para los algoritmos.

La importancia de restringir correctamente las entradas.

Relación con la clase

Se relaciona con la definición de problemas bien planteados.

Los conceptos de función característica donde se asigna un número y tiene una condición.

6.2. Ejemplo 2

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

RESTRICCIONES: El número no puede ser cero.

DATOS DE ENTRADA: El conjunto de datos de entrada E está compuesto por el conjunto de los números reales, excepto el cero.

$$E \subset R1, \quad num \in E \; de \; (-\infty, \infty) - \{0\}$$

NOTA: R1 representa al conjunto de números reales de una dimensión.

DATOS DE SALIDA: El conjunto de salida S está compuesto por dos valores mutuamente excluyentes.

Un posible conjunto de salida son los valores enteros 0 o 1, donde 0 indica que el valor es positivo y 1 indica el valor es negativo.

$$res = 0, \; si \; num \in (0, \infty), \quad res = 1, \; si \; num \in (-\infty, 0)$$

El análisis de este algoritmo muestra que su estructura es sumamente simple y eficiente, ya que se basa en una sola comparación lógica para clasificar el número de entrada como positivo o negativo, excluyendo el cero como restricción principal. Esta simplicidad le otorga una complejidad temporal O(1) y una complejidad espacial O(1), lo que lo convierte en un algoritmo óptimo para este tipo de problema, sin importar la magnitud del número ingresado.

El dominio de entrada son los números reales distintos a cero.

La salida también es binaria.

La restricción del cero muestra que no todos los valores son válidos en el algoritmo.

Aprendizaje

Trabajar con intervalos de números reales en lugar de enteros, donde se amplia el dominio del problema.

La importancia de excluir o tratar casos especiales por ejemplo el infinito.

Comprender los problemas simples los cuales se deben pensar las condiciones de entrada para evitar algún mal resultado.

Relación con la clase

Refuerza el concepto de entrada y salida.

La restricción del problema.

Estructuras condicionales.