Algoritmos

Objetivos

- Conocer el concepto de algoritmo y proceso.
- Conocer la estructura básica de un programa.
- Distinguir entre Información de Entrada y de Salida.
- Conocer y poner en práctica las formas de representación de un algoritmo
- Resolver algoritmos sencillos.
- Interpretar el enunciado de un problema
- Predecir el resultado a obtener en función de las acciones tomadas.
- Manipular herramientas informáticas para resolver problemas.
- Conocer las diferentes formas de almacenamiento de datos.
- Conocer y poner en práctica los diferentes tipos de datos.
- Realizar las acciones correctas de acuerdo al tipo de dato
- Conocer y poner en práctica el uso de variables y constantes.
- Reconocer los distintos tipos de errores.
- Conocer y poner en práctica operadores aritméticos y de relación.

Contenidos

Proceso. Algoritmos. Características de un algoritmo. Lenguaje algorítmico, Información de Entrada, Información de Salida. Uso de Variables Solicitar información de entrada y mostrar resultados en la salida.

Análisis de enunciados. Interpretación de pre y post condición. Análisis de datos de entrada y de salida. Restricciones. Errores. Comentarios.

¿Qué es un problema?

En esta unidad vamos a presentarles una metodología de **Análisis de Problemas** que les permitirá identificar los componentes que forman un problema y comprender qué es lo que hay que resolver, qué datos se necesitan, con qué valores y cuáles serán las salidas y los resultados del problema.

En esta unidad vamos a presentarles una metodología de **Análisis de Problemas** que les permitirá identificar los componentes que forman un problema y comprender qué es lo

que hay que resolver, qué datos se necesitan, con
qué valores y cuáles serán las salidas y los resultados del problema.

Comenzamos con los conceptos básicos, y es por eso que nos preguntamos:

Podemos decir que un problema **es el planteamiento de una situación a resolver mediante la aplicación de algún método**.

En la vida cotidiana resolvemos muchos problemas sin darnos cuenta y de forma automatizada pero también lo hacemos de forma innovadora o creativa.

En esta materia nos ocuparemos de problemas que se resuelven programando una computadora y que se describen por medio de enunciados.

Debemos entonces hacer una diferencia entre problema y enunciado.

Los enunciados son la descripción de un problema que se puede resolver mediante una computadora. Es decir, **los problemas se plantean mediante enunciados**.

En los enunciados debería incluirse toda la información necesaria para poder resolver el problema.

Volviendo a los problemas generales, ante un problema, lo que debemos hacer es resolverlo, utilizando un método propio o planeamiento para encontrar una solución.

¿Cómo empezar a resolver un problema?

Para poder resolver un problema, la primera cuestión a plantearse parece bastante evidente. Pero esta "evidencia" es lo que hace que en la mayoría de los casos se pase por alto, o bien se realice muy rápido, con la consecuencia de traernos inconvenientes futuros a lo largo de toda la resolución del problema.

Entonces, debemos enfocarnos en:

Comprender el problema

Comprender un problema implica "saber" cuál es el problema. Así, si no sabemos cuál es el problema ni la información para resolverlo, difícilmente podamos llegar a una solución adecuada.

Esto que resulta tan simple decirlo, no siempre se realiza de la forma apropiada. Por ello, es que nos planteamos una **metodología** para poder determinar un orden y así realizar el análisis del problema.

¿Cómo podemos realizar un adecuado análisis?

Para comprender un problema debemos realizar lo que se denomina *Análisis del Problema (también llamado Análisis de Enunciados).*

Pasamos a explicar cómo podemos realizar un adecuado análisis de un problema...

Análisis del problema

Consideremos un problema que se nos presenta descripto en un enunciado.

Entonces, el primer paso será **leer el enunciado del problema**. Ahora bien, ¿cuántas veces debe leerse un problema? La cantidad de veces que sea necesario. Seguramente, no alcanzará con leer el enunciado una sola vez, ni otra más, recién en una tercera lectura podremos empezar a comprender el problema con todos sus datos.

La lectura tampoco finalizará acá, ya que será necesario volver a releer el enunciado ante cuestiones, dudas o situaciones que no se puedan definir en las siguientes etapas.

Este primer paso de una lectura reiterada del problema, los programadores muchas veces la suelen pasar por alto, o bien, leen el enunciado "por arriba" comenzando rápidamente a escribir el programa.
Nuestra experiencia nos hizo ver que hay que tomarse todo el tiempo necesario en el momento del **análisis del problema**, porque los errores arrastrados serán luego más costosos de resolver.

Proponemos como ejemplo de un **problema**: un viaje de vacaciones.

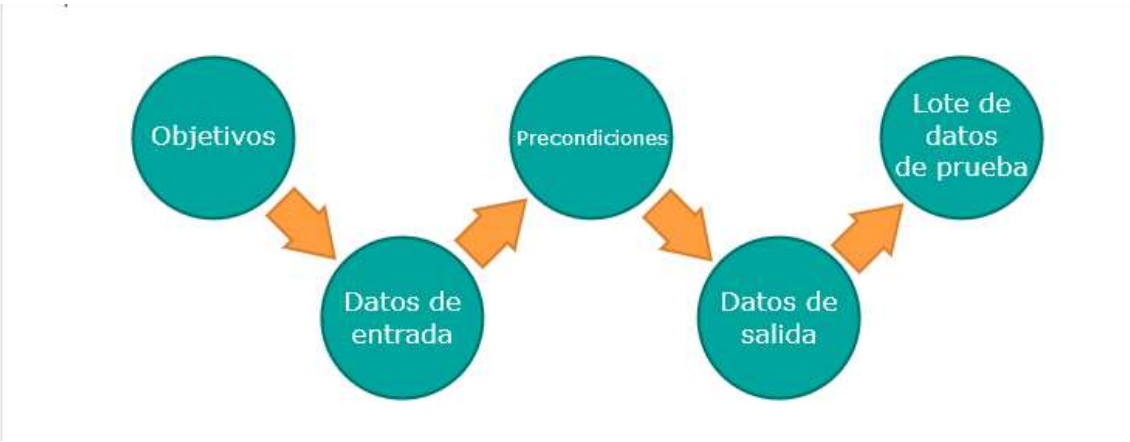
Una de las primeras cuestiones que nos planteamos es el propósito del viaje. Si bien un viaje de vacaciones, es de esperar que sean para descansar, podemos tener varios propósitos como ser: conocer un lugar distinto, ir a una playa, disfrutar de sierras o montañas, practicar un deporte como esquiar, compartir unos días con familiares o amigos, etc.

Es decir, que el **problema tiene un propósito, un OBJETIVO.** Este objetivo del problema será el primero de los componentes que debemos identificar.

Veamos a continuación cuáles son todas las componentes que debemos identificar de un problema...

Componentes de un problema

Como hemos dicho, el objetivo del problema es el primero de los componentes que debemos identificar en un problema para luego continuar con los demás componentes, que se muestran en el siguiente esquema:



Describimos a continuación cada uno de los componentes...

1. Objetivos

Como hemos mencionado, el objetivo es el primero de los componentes que debemos identificar en un problema.

El **objetivo** explica el problema a resolver, lo que hay que realizar. Es importante destacar que el objetivo indica el **QUÉ** y no el **CÓMO**. En el objetivo se plantea lo que se tiene que realizar, pero no de qué manera se va a resolver.

¿Cómo definimos un objetivo?

- El objetivo, es un párrafo en el que se tiene que explicar de manera coloquial desde dónde se parte y hacia dónde se debe llegar.
- El objetivo explica el problema a resolver, lo que se tiene que realizar.

Retomando nuestro ejemplo del viaje de vacaciones...

Una vez planteado el objetivo del viaje, llegarán otras preguntas como, por ejemplo, qué presupuesto disponemos, cuántos días durará el viaje, si viajaremos en algún medio de transporte en particular, quiénes van a viajar, etc.

Todos estos datos representan la información con la que contamos para llegar a una solución, es decir que, de acuerdo a las respuestas a estos DATOS, serán las vacaciones.

Entonces, luego de tener bien en claro qué es lo que se tiene que hacer, el paso siguiente es identificar la información con la que se cuenta para poder lograr el objetivo. Esto se realizará especificando los **Datos de Entrada.**

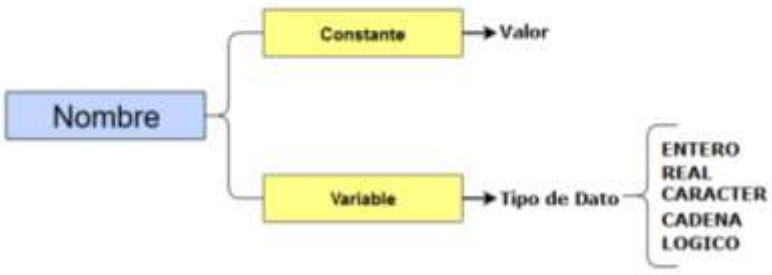
2. Datos de entrada

Los Datos de Entrada especifican la información que se tiene antes de comenzar a resolver el problema.
Son los datos con los que se cuenta para poder llegar a una solución.

Las **características** que se deben especificar para cada dato de entrada son:

- Nombre.
- Constantes y variables.
- Tipos de datos.

En el siguiente esquema presentamos cómo es la relación de los elementos de cada Dato de Entrada, y que a continuación explicaremos con mayores detalles:



2.1. Nombre

A cada Dato de Entrada le vamos a definir un NOMBRE. El nombre debe ser representativo del dato para no tener confusiones sobre su valor. Por ejemplo, si el dato va a representar la **Edad de una persona**, lo ideal es que el dato se llame EDAD.

Una de las buenas prácticas de programación establece que el nombre del dato debe representar su valor.
Es importante evitar los nombres genéricos, que muchas veces son muy tentadores, como N1, N2, Dato, NÚMERO, X, etc. , ya que dificultan la lectura y comprensión del problema.

Volviendo a nuestro ejemplo del viaje de vacaciones...

Los **datos de entrada** podrían ser: PRESUPUESTO, DÍAS DE DURACIÓN, MEDIO DE TRANSPORTE.

Suponiendo que te gusta manejar y disfrutar de un viaje en auto, ¿Vas a destinar el mismo presupuesto y la misma cantidad de días para todas las vacaciones?
Seguramente que no, ya que más allá de la variación de precios, el presupuesto seguramente será distinto. Lo mismo sucederá con la cantidad de días, en algún viaje le podrás destinar un fin de semana largo, 2 días, o en otro viaje podrás destinar 14 días. En cambio, siempre viajarás en auto, ya que te gusta y disfrutás del manejo.

Llegamos a la conclusión que el medio de transporte, uno de los datos de entrada identificados, permanece siempre igual (es su auto), por lo tanto, será un dato CONSTANTE, ya que para todos los viajes será el mismo.
Por otro lado, los otros dos datos de entrada identificados, presupuesto y días de duración, serán datos VARIABLES ya que serán distintos para cada viaje.

Resumiendo:

DATO	TIPO
MEDIO DE TRANSPORTE	CONSTANTE
PRESUPUESTO	VARIABLE
DURACIÓN	VARIABLE

Esto que acabamos de realizar es una clasificación de los datos de entrada en VARIABLES y CONSTANTES, y que desarrollaremos en la siguiente sección.

2.2. Constantes y variables

Además de indicar el nombre, debemos realizar una clasificación adicional por cada Dato de entrada, y será de acuerdo a si el valor del dato va a ser el mismo o si puede cambiar durante la ejecución del programa.

Si el dato tiene siempre el mismo valor para todos los problemas, entonces el dato de entrada es **CONSTANTE** si en cambio, su valor puede ser distinto para cada problema o bien puede cambiar dentro del mismo problema, el dato de entrada es **VARIABLE.**

Las constantes matemáticas son ejemplos de **datos Constantes**, como, por ejemplo, el número PI.Mientras que **datos Variables** pueden ser los datos de una factura (fecha, productos, importe), ya que lo usual es que los datos de cada factura sean distintos.

Para tener en cuenta:

Los Datos de Entrada Constantes deben indicarse con su valor.

Por ejemplo, **PI = 3,14** o **IVA = 0,21.**

Siempre debe haber, por lo menos, un dato de entrada **Variable**.

Volvamos a nuestro ejemplo de las vacaciones y los datos de entrada que definimos....

Sabemos que los **DÍAS DE DURACIÓN**, es un dato VARIABLE. Analizando un poco más en profundidad, qué valores posibles podrá tener, como se trata de días, el valor será un número. Si queremos ser más específicos, debería ser un NÚMERO NATURAL.
Del mismo modo, el otro dato variable, el **PRESUPUESTO**, también será un número, pero en este caso, el número podrá ser un NÚMERO REAL.

Esta especificación que acabamos de mencionar con los datos corresponde a lo que llamaremos TIPOS DE DATOS. Los mismos, serán desarrollados en el siguiente apartado.

2.3. Tipos de datos

Ya sabemos que **el Tipo de Dato indica cuál es el conjunto de valores posibles del dato**. Además, permite definir cuáles serán las operaciones que se podrán aplicar a este dato.

El tipo de dato se debe especificar para los Datos de Entrada Variables, ya que en los Constantes se deducen de acuerdo a su valor.

Vimos también que los tipos de datos pueden ser números (enteros, reales, por ejemplo). Pero esos no son los únicos. Se estarán imaginando que algunos problemas necesitan trabajar con nombres de personas, letras o símbolos para indicar algún código, y hasta con valores de tipo SI-NO.

Es por eso que en la etapa del Análisis del problema vamos a trabajar con los siguientes tipos de datos estándar:

Tipo de Dato	Valor
Entero (Integer)	Números enteros
Real (Float)	Números reales
Caracter (Character)	Un solo símbolo (letra, dígito, signo, etc.)
Cadena (String)	Un conjunto de caracteres (palabra, frase, etc.)
Lógico (Boolean)	Verdadero o Falso

Las operaciones sobre cada tipo de datos las veremos más adelante, ya que por ahora solo interesa saber los tipos de datos y sus posibles valores.

Retomando nuevamente el ejemplo del viaje de vacaciones...

Sabemos que DÍAS DE DURACIÓN es un dato VARIABLE de tipo ENTERO, pero...¿cualquier valor dentro de los números enteros es correcto?

Sabemos que por lo menos podemos tomarnos 1 día de vacaciones. Esa será la cantidad mínima de días que se podrá viajar, no tiene sentido irse 0 días o -5 días de viaje. Por lo tanto, solo serán válidos algunos números enteros. En este caso podemos pensar en un subconjunto que está formado por los números enteros mayores o iguales a 1.

Por otro lado, si bien tener muchos días de vacaciones sería algo que deseamos todos, nuestras obligaciones (y nuestro jefe) nos limitarán la cantidad máxima de días ¿Cuál será? ¿30, 60, 28? Será un aspecto a definir con mayor profundidad en las próximas unidades, pero sabemos que existe un límite máximo en los datos.

Estos límites o subconjuntos que acabamos de indicar sobre los TIPOS de DATOS es lo que llamamos PRECONDICIONES o, también, RESTRICCIONES, y lo veremos en la próxima sección.

3. Precondiciones

Con las **Precondiciones** se indican los valores posibles que deberán tener los Datos de Entrada Variables.

Acá tenemos que prestar mucha atención para poder diferenciar el Tipo de dato de las precondiciones, ya que son conceptos similares, aunque indican cosas distintas.

> El **tipo de dato** indica el conjunto posible de valores, por ejemplo, los números reales, mientras que las **precondiciones** establecen cuáles de esos números reales serán válidos para el problema.

Por ejemplo, ante un dato de entrada que indica la longitud del lado de un cuadrado, el tipo de dato será REAL. Ahora bien, ¿cualquier número REAL es válido para la longitud de un lado?

Por supuesto que no, ya que para que sea lado, deberá ser un número mayor a 0.

Esa es la precondición.

Las precondiciones las podemos especificar como condiciones matemáticas o expresarlas en palabras, siempre y cuando no haya ambigüedades y quede bien claro cuáles son los valores posibles.

Por ejemplo, la **precondición de la longitud del lado de un cuadrado**, se puede expresar de la siguiente manera:

```
LADO > 0

O también:

LADO tiene que ser mayor a 0.
```

Las precondiciones son importantes porque indican cuándo un dato es un dato válido para el problema. Así, tenemos que tener en cuenta que: **un dato de entrada será válido cuando cumpla con todas las Precondiciones**.

4. Datos de salida

Hasta el momento hemos especificado todo lo relacionado con los datos que necesitamos para resolver el problema, es decir la ENTRADA del problema. Así como se

indica la entrada, se debe especificar los datos que se generarán al resolver el problema, que se llaman **Datos de Salida.**

Los **Datos de salida** se utilizan para especificar la información que se tiene después de resolver el problema.

Para este tipo de datos no será necesario realizar ninguna otra clasificación porque es de esperar que todos los datos de salida sean siempre VARIABLES, pues dependerán de los Datos de Entrada.

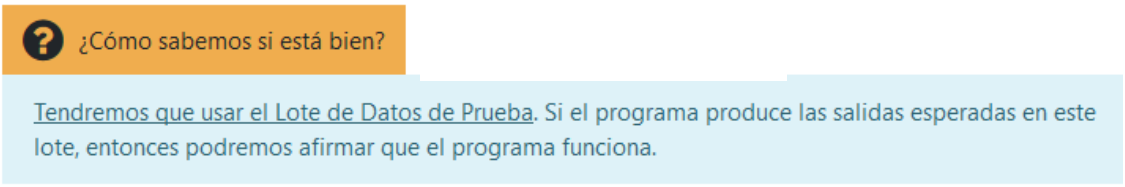
Es decir, que solo alcanza con definir la lista de nombres de los Datos de Salida.

5. Lote de Datos de Prueba

Hemos analizado y especificado la Entrada y la Salida del problema, para luego plantear un *conjunto de datos de prueba*. Este lote de datos servirá para más adelante, luego de escribir el programa, para poder realizar lo que se denomina *Prueba de Escritorio*.

Una forma de verificar que el programa no tenga errores lógicos es mediante la *realización de la Prueba de Escritorio*, tomando como datos de entrada los especificados en este **Lote de Datos de Prueba.**

Una vez desarrollado el programa, nos preguntamos:



El **lote de datos de prueba** se compone de *una lista de valores para los datos de entrada y las salidas correspondientes* a esos datos. Podemos representarlo en forma de tabla de valores, sin necesidad de especificar cómo se llegó a ese resultado, ya que de eso se ocupará el programa.

ENTRADAS				**SALIDAS**		
Dato Entrada 1	Dato Entrada 2	...	Dato Entrada N	Dato Salida 1	...	Dato Salida N
-	-	-	-	-	-	-
-	-	-	-	-	-	-

Armar el lote de Datos no es una tarea para tomar a la ligera, ya que serán los datos con los que se probará el programa. Si el lote es erróneo o hay situaciones que no se

contemplan, entonces la prueba será incompleta y no se sabrá si el programa funcionará correctamente para todas las situaciones.

Veamos un ejemplo:

Tenemos un programa que realiza la suma de dos números si ambos son positivos o realiza la resta si ambos son negativos.

El lote de datos de prueba deberá abarcar las dos situaciones, ambos números negativos o ambos números positivos. No tiene sentido realizar una tabla de cientos de valores, ambos positivos, ya que todos contemplan una misma situación y no se está evaluando cuando ambos números son negativos. Para este caso, un lote de datos ideal estaría formado por un par de números positivos y otro par de números negativos.
Por ejemplo, podemos probar con los números -2;-3;-5; 7; 8; 15; 1000.

Operadores Aritméticos

Son los operadores fundamentales de la aritmética con el agregado de la potencia y el módulo o residuo, que devuelve el resto entero que se produce al realizar un cociente entre dos números enteros.

Operador	Nombre	Ejemplo	Resultado	Descripción
+	Suma	12 + 3	15	Devuelve la suma de dos expresiones.
-	Resta	12 - 3	9	Devuelve la resta de dos expresiones.
*	Multiplicación	12 * 3	36	Devuelve el producto de dos expresiones.
/	División	12 / 3	4	Devuelve el cociente de dos expresiones.
^	Potenciación	12 ^ 3	1728	Devuelve la potencia entre dos expresiones.
%	Módulo o Residuo	12 % 3	0	Devuelve el resto del cociente entre dos enteros.

Siguiendo los fundamentos de la aritmética, los operadores de multiplicación, división y módulo tienen mayor prioridad que la suma y resta. Así, por ejemplo, la expresión 2 + 3 * 4 devuelve como resultado 14, pues primero se evalúa el término 3 * 4 que resulta 12, y luego se realiza la suma entre 2 y 12.

Operador de asignación

Hasta ahora hemos visto como declarar variables, las reglas de nombrado y la convención más difundida. Pero recordá que hasta ahora las variables no están inicializadas, es decir, su valor es indefinido.

El operador de asignación de PSeInt es el =, aunque por defecto también puede usarse el <-, que es compuesto, dado que se conforma de dos caracteres. Es muy importante que no dejes un espacio entre ellos.

La sintaxis para asignar un valor a una variable es la siguiente:

<variable> = <expresión>;

La palabra **<variable>** indica que en ese lugar debe ir el identificador de una variable definida.

= es el operador de asignación.

La palabra **<expresión>** indica que en ese lugar debe ir una expresión válida, de un tipo de dato compatible con lo que espera guardar la variable.

El **;** es obligatorio e indica que finalizó la instrucción.