TEMA 1
INTRODUCCIÓN
ALGORITMOS Y PROGRAMAS

1. Sistemas de procesamiento de la información.
2. Algoritmos: Características y diseño.
3. Ciclo de vida de una aplicación.
4. Errores.
5. Documentación de los programas.
6. Elementos básicos de un programa.
7. Estructura general de un programa.
8. Escritura de algoritmos.

1.1. - SISTEMAS DE PROCESAMIENTO DE LA INFORMACIÓN

- Es el que transforma datos brutos en información organizada, significativa y útil.

Entrada = datos

Procesador

Salida =información

Algoritmos: Características y diseño

Un algoritmo es un método para resolver un problema.

Definimos algoritmo como el conjunto de instrucciones que especifican **la secuencia de operaciones a realizar en orden**, para resolver un determinado problema .

La resolución de un problema exige:

- Definición y análisis del problema
- Diseño del algoritmo
- Transformación del algoritmo en un programa
- Ejecución y validación del problema

Cuando el procesador es un ordenador, el algoritmo ha de expresarse de una forma que recibe el nombre de programa.

Algoritmos: Características y diseño

Un programa se escribe en un lenguaje de programación y a la actividad de **expresar un algoritmo en forma de programa se le denomina programación**.

Para llegar a la realización de un programa es necesario el diseño previo de un algoritmo, de modo que **sin algoritmo no puede existir un programa**.

Los algoritmos **son independientes** del lenguaje de programación en que se expresan.

Un lenguaje de programación es sólo un medio para expresar un algoritmo y un ordenador es sólo un procesador para ejecutarlo.

El diseño de la mayoría de los algoritmos requiere creatividad y conocimientos profundos de la técnica de programación

Algoritmos: Características y diseño

Características de los algoritmos

- Debe ser preciso e indicar el orden de realización de cada paso.
- Estar **definido**. Si se sigue un algoritmo dos veces se debe obtener el mismo resultado.
- Debe ser **finito**. Si se sigue un algoritmo se debe terminar en algún momento.
- La definición de un algoritmo debe describir tres partes:
  - Entrada.
  - Proceso.
  - Salida.
- Para un determinado problema se pueden construir diferentes algoritmos de resolución.

Algoritmos: Características y diseño

- La elección del más adecuado se basa en una serie de requisitos de calidad que adquieren importancia a la hora de evaluar el coste de su diseño y mantenimiento.

Las características que debe cumplir un programa son:

Legibilidad.- Ha de ser claro y sencillo de tal forma que facilite su lectura y comprensión.

Fiabilidad.- Ha de ser robusto, es decir capaz de recuperarse frente a errores o usos inadecuados.

Portabilidad.-Su diseño debe permitir la codificación en diferentes lenguajes de programación.

Modificabilidad.- Ha de facilitar su mantenimiento, es decir, poder hacer las modificaciones y actualizaciones necesarias para adaptarlo a una nueva situación.

Algoritmos: Características y diseño

		Definición del problema
Análisis del problema		Especificaciones de entrada
		Especificaciones de salida
Diseño del algoritmo		Herramientas de programación
		Codificación del programas
Resolución con ordenador		Ejecución del programa
		Comprobación del programa

Algoritmos: Características y diseño

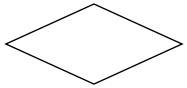
- Para representar un algoritmo debemos utilizar un método que permita independizar dicho algoritmo del lenguaje de programación elegido, permitiendo que pueda ser codificado en cualquier lenguaje. **Los métodos usuales para representar un algoritmo son :**
  - **Diagrama de flujo**
  - ***Pseudocódigo***

- **Diagrama de flujo.-** Es una de las técnicas de representación de algoritmos más antiguas , aunque su empleo disminuyó considerablemente con los lenguajes de programación estructurados.

  Un diagrama de flujo utiliza símbolos estándar que contienen los pasos del algoritmo escritos en esos símbolos.

Los símbolos más utilizados son :

Proceso : cualquier tipo de operación que pueda originar cambio de valor, formato, operaciones aritméticas etc.



Decisión : indica operaciones lógicas o de comparación entre datos y en función del resultado determina el camino a seguir.

Terminal : representa el comienza y el final de un programa.

Entrada / Salida: de información. Este símbolo se puede subdividir en otros de teclado, pantalla, impresora disco etc.

Dirección del flujo: (Indica el sentido de ejecución de las operaciones).

Pseudocódigo.

El pseudocódigo es un lenguaje de descripción de algoritmos, que hace más fácil el paso final de codificación .

Nace como medio de representar las estructuras de control de programación estructurada.

Ventajas

- Se puede modificar fácilmente si detecta un error en la lógica del programa, y puede ser traducido fácilmente a los lenguajes estructurados.
- Utiliza para representar las acciones sucesivas, palabras reservadas, y exige la indentación -sangría en el margen izquierdo- de diferentes líneas.

Las líneas precedidas por **// se denominan comentarios**, es decir una información al lector del programa que no realiza ninguna instrucción ejecutable.

Palabras reservadas del pseudocódigo:

leer, escribir

si <condición> **entonces**
<**acción**>
si_no
<**acción**>
fin_si

mientras <**condición** >
<**acción**>
fin_mientras

para valor-inicial **hasta** valor-final
<**acción**>
fin_para

repetir
<**acción**>
hasta_que <**condición**>

2.- Algoritmos: Características y diseño

- El conjunto de métodos y técnicas que ayudan al desarrollo de programas que cumplan estos requisitos, se denomina **Metodología de la Programación.** Las técnicas que vamos a estudiar son:

  - **Programación modular.** Se basa en la realización de una serie de descomposiciones sucesivas del algoritmo principal.

    El programa quedará formado por una serie de **módulos,** cada uno de los cuales realiza una parte de la tarea total.

  - **Programación estructurada.** Se basa en el uso exclusivo de las estructuras **secuencial, alternativa y repetitiva** , para el control del flujo de ejecución de las instrucciones. Los programas así diseñados serán fáciles de verificar, depurar y mantener.

  - **Programación orientada a objetos**. Es una evolución lógica de la programación estructurada, en la que el concepto de variable local se hace extensible a los propios subprogramas que acceden a estas variables.

3 CICLO DE VIDA DE UNA APLICACIÓN INFORMÁTICA

Una aplicación, es un **conjunto de uno o más programas** interrelacionados , que tienen por objeto la realización de una determinada tarea mediante el uso de un sistema informático.

El ciclo de vida de una aplicación *informática* es el **proceso** que se sigue desde el **planteamiento de un problema hasta que se tiene una solución** instalada en el ordenador y en funcionamiento por el usuario, mientras esta sea de utilidad.

3 CICLO DE VIDA DE UNA APLICACIÓN INFORMÁTICA

Se compone de varias fases que agrupamos en dos bloques

	Análisis	Especificaciones E / S
Fase de Diseño	**Programación**	Algoritmo
	Codificación	Programa
	Edición	Programa Fuente
	Compilación	Programa objeto
Fase de Instalación	**Montaje**	Programa ejecutable
	Prueba de ejecución	Aplicación
	Explotación y Mantenimiento	

4.- ERRORES

Según el momento en que se detectan pueden ser:

- De compilación — Sintaxis
- De ejecución — Operaciones no permitidas
- De lógica — Resultados no correctos.

  Los más difíciles de corregir.
- De especificación — Especificaciones incorrectas,
  - son los más costosos -.

5. Documentación de los programas

Con el fin de facilitar la explotación y el mantenimiento de un programa es fundamental que este se acompañe de una documentación clara, amplia y precisa.

Existen dos clases de documentación según su ubicación:

- **Documentación interna**
- **Documentación externa**

5. Documentación de los programas

- **Documentación interna:**
  - ***Comentarios***. Son frases explicativas que se insertan en cualquier lugar del programa fuente y son ignoradas por el compilador.
  - ***Código autodocumentado***. Las palabras reservadas de los lenguajes constituyen en si mismas, parte de la documentación.

    Se puede mejorar la documentación de un programa :
    - Utilizando identificadores adecuados para nombrar variables, constantes, subprogramas etc.
    - Declarando constantes para valores fijos.
    - Sangrando, paginando e intercalando líneas en blanco para facilitar la lectura del programa.

Documentación externa.

Es el conjunto de documentos que acompañan al programa, sin formar parte de el.

Debe incluir al menos:

- Especificaciones del análisis.
- Descripción del diseño del programa.
- Descripción del programa principal y subprogramas.
- Manual de usuario.
- Manual de mantenimiento.

Elementos básicos de un programa

Son objetos de un programa todos aquellos manipulados por las instrucciones.

Todo objeto tiene tres atributos:

- **Nombre**.- Es el identificador del mismo.
- **Tipo**.- Conjunto de valores que puede tomar.
- **Valor**.- Elemento del tipo que se le asigna.

Los elementos básicos de un programa o algoritmo

- ***Palabras reservadas*** (inicio, fin, si -entonces . . ., *etc.).*
- **Identificadores** (nombres de variables esencialmente).
- ***Caracteres especiales*** (coma, apóstrofo, etc.).
- **Constantes.**
- ***Variables.***
- **Expresiones.**
- **Instrucciones.**

TIPOS DE DATOS SENCILLOS

Identificadores.

Son palabras creadas por el programador para dar nombre a los objetos y demás elementos que necesitamos declarar en un programa, como variables, estructuras de dato, archivos, subprogramas, etc.

- En general se utiliza una cadena de letras y dígitos .
- Un dato es la expresión general que describe los objetos con los cuales opera un ordenador.

En el proceso de solución de problemas, el diseño de las estructuras de datos es tan importante como el diseño del algoritmo y del programa.

TIPOS DE DATOS

Tipos de Datos

- **Simples**
  - **Numéricos** (integer, real).
  - **Lógicos** (boolean).
  - **Carácter** (char, string).
- **Estructuras de datos**
  - **Internas**
    - **Estáticas (tablas)**
    - **Dinámicas** (Listas, Pilas, Colas ,Árboles)
  - **Externas**
    - **Bases de Datos**
    - **Ficheros**

Datos numéricos

El tipo numérico es el conjunto de los valores numéricos.

Se puede representar de dos formas distintas:

- Tipo numérico entero (int) es un subconjunto finito de los números enteros. No tienen componentes fraccionarios o decimales y pueden ser positivos o negativos. Se denominan en ocasiones números de coma fija
- Tipo numérico real (float) consiste en un subconjunto de los números reales. El número real consta de una parte entera y una parte decimal y pueden ser positivos o negativos.
- Datos lógicos (booleanos) El tipo lógico es aquel dato que sólo puede tomar uno de estos valores: verdadero (true) o falso (false). Este tipo se utiliza para representar las alternativas (si/no) a determinadas condiciones.

Datos tipo carácter y tipo cadena

El tipo carácter es el conjunto finito y ordenado de caracteres que reconoce el ordenador. Un dato tipo carácter contiene un único carácter.

Los caracteres que reconocen los diferentes ordenadores no son estándar, sin embargo la mayoría reconocen los siguientes caracteres alfabéticos y numéricos:

- ✓ **Caracteres alfabéticos (A….Z) (a..z)**
- ✓ **Caracteres numéricos (1,2…..0)**
- ✓ **Caracteres especiales (+, -, *, /, ^, ., ;, >, <, $ …)**

Una cadena (string) de caracteres es una sucesión de caracteres que se encuentran delimitados por una comillas simples o dobles , según el lenguaje de programación.

Constantes y variables

Los programas contienen ciertos valores que no deben cambiar durante la ejecución del programa, estos valores se llaman constantes.

- La mayoría de lenguajes de programación permiten diferentes tipos de constantes: enteras, reales, caracteres y boolean o lógicas.

- *Una constante tipo carácter* o constante de caracteres consiste en un carácter válido encerrado entre apóstrofos.

Una variable es un objeto cuyo valor puede cambiar durante el desarrollo del algoritmo o ejecución del programa. Hay diferentes tipos: enteras, reales, caracteres , boolean o lógicas y cadena.

- Una variable se identifica por los siguientes atributos: nombre y tipo.

  Los nombres a veces conocidos como identificadores suelen constar de varios caracteres alfanuméricos de los cuales el primero normalmente es una letra.

- Los nombres de las variables elegidas para el algoritmo deben se significativas y tener relación con el objeto que representan.

Expresiones

- Las expresiones son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales.
- Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones indicadas.
- Una expresión consta de **operandos y operadores**, y según sea el tipo de objetos que manipulan, las expresiones se clasifican en :
    - Aritméticas
    - Lógicas
    - Carácter

Expresiones aritméticas.

- Son análogas a las fórmulas matemáticas. Las variables y constantes son numéricas (enteras o reales) y las operaciones son aritméticas.

operadores:

+ suma

\- resta y cambio de signo

* producto

/ cociente

** ^ potencia

div división entera

mod módulo o resto.

Operadores div y mod

El operador **div** en algunos lenguajes representa la **división entera**.

Ejemplo : A div B

- Sólo se puede realizar si A y B son variables enteras.

19 div 6 = 3	19 mod 3 = 1
15 div 6 = 2	15 mod 6 = 3

Reglas de prioridad

- **Las expresiones que tienen dos o más operandos requieren unas reglas matemáticas** que determinen el orden de las operaciones, estas reglas de prioridad son las siguientes.
  - Las operaciones encerradas entre paréntesis se evalúan primero. Si existen diferentes paréntesis anidados , las expresiones más internas se evalúan primero.
  - Las operaciones aritméticas dentro de una expresión suelen seguir el siguiente orden:
    - operador exponencial (^ ↑ **)
    - producto y cociente * /
    - operadores div y mod
    - operador suma y resta
  - Cuando varios operadores de igual prioridad coinciden en una expresión, el orden de prioridad es de izquierda a derecha.

Ejemplos

Ejemplo 1.2 Calcula el resultado de las siguientes expresiones

3 + 6 * 14	(87)
8 + 7 * 3 + 4 * 6	(53)
- 4 * 7 + 2 ^ 3 / 4 - 5	(-31)

Ejemplo 1.3 Convertir en expresiones aritméticas las siguientes expresiones algebraicas:

$a^2 + b^2$	a ^2 + b ^ 2
$\frac{x + y}{u + \frac{w}{a}}$	(x + y) / (u + w / a)

Ejemplo 1.4 Evaluar la expresión

12 + 3 * 7 + 5 * 4	(53)

Expresiones lógicas.

- Sólo pueden tomar dos valores verdadero y falso.

- Se forman combinando constantes lógicas, variables lógicas y otras expresiones lógicas, utilizando los operadores lógicos (**not , and , or**) y los operadores de relación.

Operadores lógicos

La definición de los operadores lógicos se resume en tablas conocidas como tablas de verdad:

Ejemplos

a	No a
verdad	falso
Falso	verdad

No (6 > 10) es verdad

a y b son verdad sólo si **a** es verdad **y b** es verdad

a	b	a Y b
verdad	verdad	verdad
verdad	falso	falso
falso	verdad	falso
falso	falso	falso

a o b son verdad cuando **a, b** , o ambas son verdad

a	b	a O b
verdad	verdad	verdad
verdad	falso	verdad
falso	verdad	verdad
falso	falso	falso

Operadores de relación.

= < > <= >= <>

- Realizan comparaciones de valores de tipo numérico o carácter.
- Se utilizan para expresar las condiciones en los algoritmos, y el resultado de la operación será verdadero o falso.
- Se pueden aplicar a cualquiera de los cuatro tipos de datos, enteros, reales, lógicos y carácter.
- Cuando se utilizan los operadores de relación con valores lógicos, la constante false (falso) es menor que la constante true (verdad)

Ejemplos : Evaluar las siguientes expresiones

- **Ejemplos 1.6** A=4 B=3

  A > B — es verdader

  (A - 2) < (B - 4) — es falso

- **Ejemplos 1.7** numero=5

  (1 < 5) and (5 < 10) — es verdad

  (5 > 10) or (`A` < `B`) — es verdad

  (numero = 1) or (7 >=4) — es verdad

Funciones internas

- Las operaciones necesarias en los programas exigen en muchas ocasiones además de las operaciones aritméticas básicas, un determinado número de operadores especiales llamadas *funciones internas* .

- Estas **funciones** utilizan argumentos reales o enteros y los resultados dependen de la tarea que realice la función.

Función	Descripción	Tipo argumento	Resultado
abs(x)	v. absoluto de x	entero o real	igual argumento
redondeo(x)	redondeo de x	real	entero
cuadrado(x)	cuadrado de x	entero o real	igual argumento
raiz2(x)	raíz cuadrada de x	entero o real	real
trunc(x)	truncamiento de x	real	entero

Ejemplos

Ejemplo 1.8

Utiliza las funciones internas para representar el algoritmo de resolución de una ecuación de segundo grado $ax^2+bx+c=0$

x1= (-b + raiz2 (cuadrado(b) - 4 * a * c)) / (2 * a)

x2= (-b - raiz2 (cuadrado(b) - 4 * a * c)) / (2 * a)