PROGRAMACIÓN Y MÉTODOS NUMÉRICOS
2503506

INTRODUCCIÓN

Andrés Agudelo
Departamento de Ingeniería Mecánica
andres.agudelos@udea.edu.co

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Información general del curso

Aulas de clase:	Miércoles 10 a.m.	→ 19 – 311
	Grupo 1: Viernes 10 a.m.	→ 20 – 238
	Grupo 2: Martes 8 a.m.	→ 20 – 341
Aula de prácticas:	Viernes 6 – 8 a.m.	→ 20 – 238

Profesor:	Andrés Agudelo
Oficina:	20 – 409
e-mail:	andres.agudelos@udea.edu.co
Horario atención:	Martes 10 a.m. – 12 p.m.

Profesor auxiliar:	David Hincapié (Estudiante instructor)
Oficina:	20 – 349
e-mail:	odavid.hincapie@udea.edu.co
Horario atención:	Lunes 4 p.m. – 6 p.m. (Oficina)

Contenido

Unidad 1

1. Programación:
   - Contexto
   - Conceptos básicos
   - Representación de algoritmos
   - Instrucciones de asignación
   - Instrucciones secuenciales
   - Tipos de variables
   - Entrada/salida de datos
   - Estructuras de decisión
   - Estructuras de repetición
   - Estructuras de datos
   - Funciones/Subprogramas
1. Introducción a Python:
   - Presentación de Python
   - Entornos de trabajo
   - Operaciones matemáticas básicas
   - Comandos más usados
   - Lenguaje de programación
   - Operaciones con arreglos
   - Construcción de gráficas
   - Programas y funciones
   - Manejo de datos y archivos
   - Interfases gráficas

Unidad 2

2. Métodos numéricos:
   - Diferenciación e integración numérica
   - Ecuaciones de una variable
   - Sistemas de ecuaciones lineales
   - Sistemas de ecuaciones no lineales
   - Ecuaciones diferenciales ordinarias: P. Valor inicial
   - Ecuaciones diferenciales ordinarias: P. Valor en la frontera
   - Ecuaciones diferenciales parciales

Metodología

- Lectura previa de los temas.
- Presentación de los temas en clase.
- Sesiones de práctica.
- Uso de pseudo-código.
- Uso de Python.
- Realización de quices y exámenes
- Trabajo en grupo: sustentación.
- Disponibilidad de notas de clase, trabajos, etc.

Página web del curso:

https://classroom.google.com/
Curso código **jwg26oq**

Evaluación

- 2 Quices → 10 %
- Examen unidad 1 → 20 %
- Prácticas de programación en Python → 20 %
- Examen unidad 2 → 20 %
- 2 trabajos unidad 2 → 30 %

Bibliografía

Unidad 1

- *Fundamentos de programación. Algoritmos, estructuras de datos y objetos.* 3^a ed. Luis Joyanes Aguilar, Mc Graw Hill, 2003.
- *Elementos esenciales para programación: Algoritmos y Estructuras de Datos.* 1^a ed. Gracia M. Gagliano y otros., Iniciativa Latinoamericana de Libros de Texto Abiertos (LATIn), 2014. http://www.proyectolatin.org/index.php/es/
- Lógica de programación. Efraín Oviedo. ECOE Ediciones, 2003
- Algoritmos estructurados. Efraín Oviedo. Fondo Editorial Cooperativo, 2001.
- Algoritmos: Conceptos básicos. César Becerra Santamaría
- Metodología de la programación. Algoritmos, diagramas de flujo y programas. Osvaldo Cairó. Alfaomega.
- Cualquier libro sobre algoritmos y programación.

Bibliografía

Unidad 1

- *Python 3 al descubierto.* 2ª ed. Fernández, A. Alfaomega, México, 2013.
- *Python 3 : Los fundamentos del lenguaje.* 2ª ed. Chazallet, S. Ediciones ENI, Barcelona, 2015.
- *Algoritmos y Programación con lenguaje Python.* Wachenchauzer, R. y otros. Autoedición, 2011. https://openlibra.com/es/book/algoritmos-y-programacion-con-lenguaje-python
- Tutorial oficial de Python 3: https://docs.python.org/3/tutorial/
- Tutorial oficial de Python 3, traducido al español: http://docs.python.org.ar/tutorial/pdfs/TutorialPython3.pdf
- Algorithmic Problem Solving with Python. https://www.eecs.wsu.edu/~schneidj/PyBook/swan.pdf
- Introducción a la programación con Python. http://www.mclibre.org/consultar/python/
- Tutorial de Python: 'Python para todos'. http://mundogeek.net/tutorial-python/
- Recursos en internet

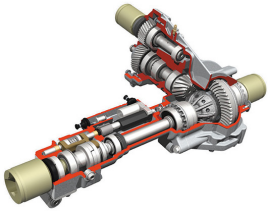
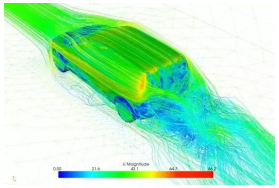
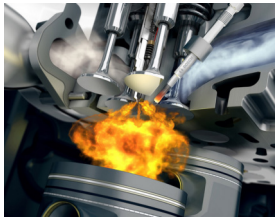
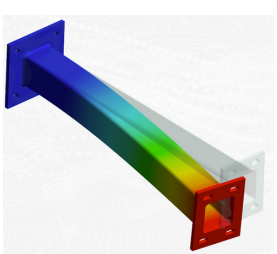
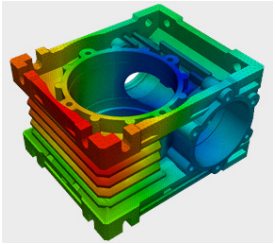
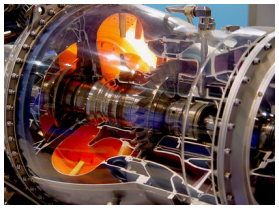
Bibliografía

Unidad 2

- *Métodos numéricos para ingenieros.* 7ª ed. Chapra, S. y Canale, R., Mc Graw Hill, 2015.
- *Análisis numérico.* 10ª ed. Burden, R. y Faires, J., Cengage Learning, 2016.
- *Iniciación a los métodos numéricos. Material didáctico, Matemáticas, No. 7.* José Antonio Ezquerro Fernández., Universidad de La Rioja, Servicio de Publicaciones, Logroño, 2012. https://dialnet.unirioja.es/descarga/libro/489813.pdf
- *Métodos numéricos* Francisco José Correa Zabala., Fondo editorial universitario, EAFIT, Medellín, 2010.
- *Introduction to Numerical Methods. Lecture notes for MATH 3311.* Jeffrey R. Cashnov. The Hong Kong University of Science and Technology, 2012. https://www.math.ust.hk/~machas/numerical-methods.pdf
- Cualquier libro sobre métodos numéricos.

Contenido

Problemas reales de ingeniería

Contenido

Informática y computadores

Informática

Francia, década de 1960: ***INFORMATIQUE***

INFORmation + Auto**MATIQUE**

"Conjunto de conocimientos científicos y técnicas que hacen posible el **tratamiento automático de la información** por medio de computadores."

Computador

"Máquina electrónica, analógica o digital, dotada de una memoria de gran capacidad y de métodos de **tratamiento de la información**, capaz de **resolver problemas** matemáticos y lógicos mediante la ejecución de **programas informáticos**."

Computadores

Computadores

Era mecánica $\leftarrow 1940 \rightarrow$ **Era electrónica**

Era mecánica

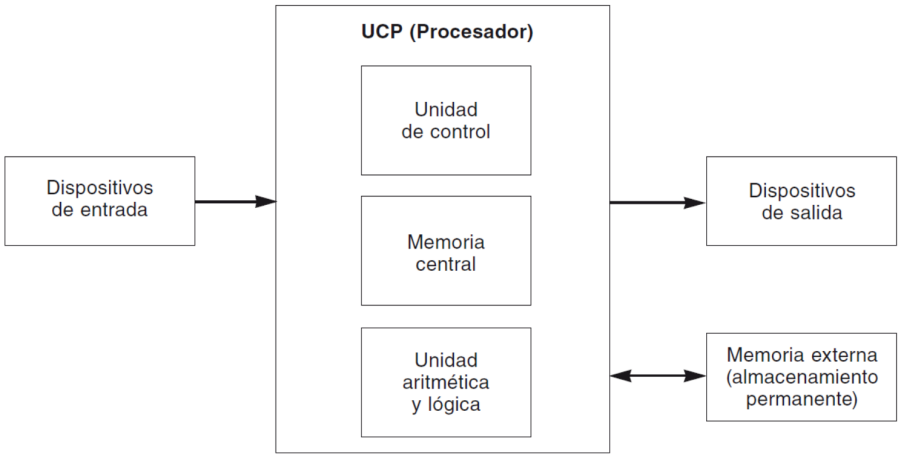
- Data desde la atigüedad $\rightarrow$ Ábaco
- Máquina diferencial de Charles Babbage (1822). Concebida para calcular funciones polinómicas. $\rightarrow$ Construida en 1999.

Computadores

Era electrónica

- 1946 → ENIAC (*Electronic Numerical Integrator And Calculator*), Universidad de Pensilvania.
- Se dividen por generaciones, comenzando por la primera (ENIAC), hasta la sexta, que está vigente en la actualidad.
  - Inicio con tubos de vacío
  - Aparición de los transistores
  - Microprocesadores
  - Reducción del tamaño, del consumo energético, y aumento de la velocidad de procesamiento.
  - Inteligencia artificial.

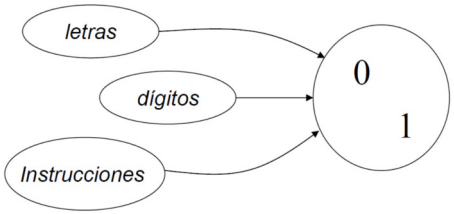
Computadores



Computadores

Codificación

En un computador toda la información (instrucciones y datos) se representa mediante cadenas de 0 y 1 (**código binario**)



Código binario

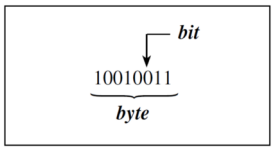
- Valores de 0 y 1 ⇒ Diferenciación de estados.
- Implementación física ⇒ Impulsos eléctricos con voltaje determinado.
  Por ejemplo: 1 → 3.3 V 0 → 0 V.

Computadores

Unidades de medida de memoria

La unidad elemental de información es el **bit** → Variable lógica que sólo admite dos valores: 0 ó 1.

8 bits → 1 **byte** (Codificación de 1 carácter)



- 1 kilobyte (kB) → 1024 (2^{10}) bytes
- 1 Megabyte (MB) → 1024 kB
- 1 Gigabyte (GB) → 1024 MB
- 1 Terabyte (TB) → 1024 GB

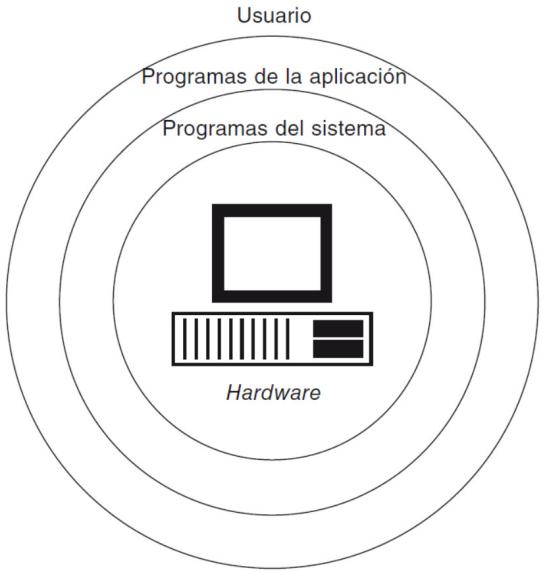
Computadores

Componentes

Hardware Componentes que integran la parte material de un computador.

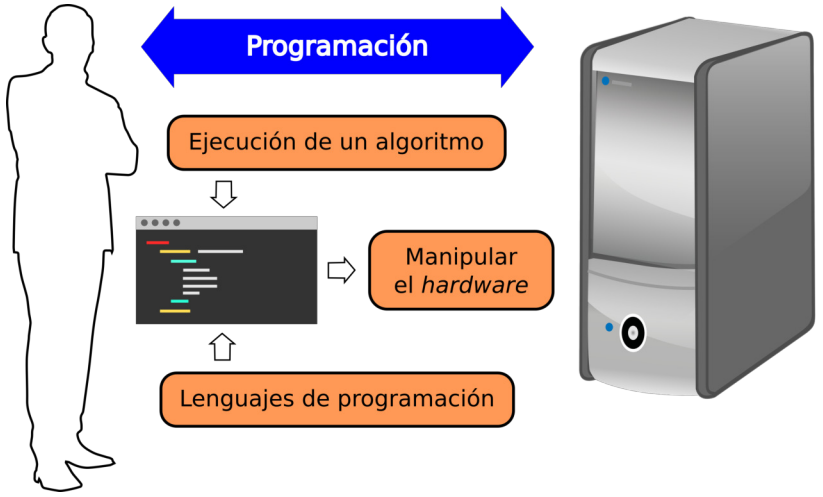
Software Programas, instrucciones y reglas informáticas para ejecutar tareas en un computador.

Software



Contenido

Programación



Programación

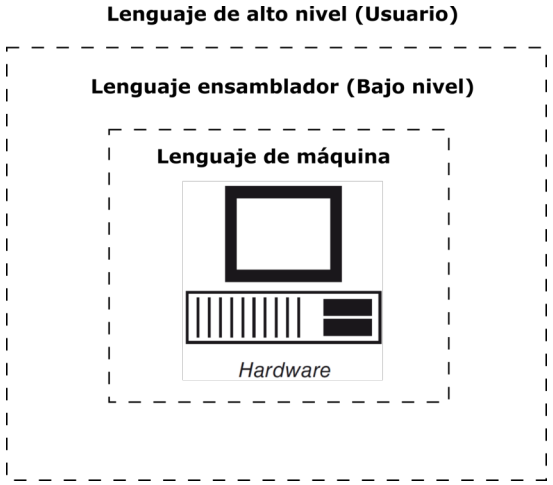
Programa

Secuencia de instrucciones que entiende el computador, y que persiguen un objetivo: resolver un problema.

Lenguajes de programación

- Lenguaje formal diseñado para la comunicación humano – máquina (entre humanos).
- Formado por un conjunto de símbolos y de reglas sintácticas (secuencia) y semánticas (significado).
- Existen diferentes clases

Lenguajes de programación



Lenguajes de programación

Lenguaje de máquina

Códigos hexadecimales que representan instrucciones, registros de la CPU, direcciones de memoria o datos.

`A0 2F`	Acceder a la celda de memoria 2F
`3E 01`	Copiar el registro 1 de la ALU (Unidad aritmética lógica)
`1D`	Sumar
`B3 31`	Guardar el resultado en la celda de memoria 31

Características

- Especializado.
- Dependiente de la máquina.
- De difícil programación.

Lenguajes de programación

Lenguaje ensamblador

Conjunto de mnemónicos que representan los códigos hexadecimales. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar **una arquitectura** de procesador. Traduce el lenguaje del usuario al lenguaje de máquina.

Características

- De nivel bajo.
- Mayor legibilidad:

```
READ 2F
REG 01
ADD
WRITE 31
```

Lenguajes de programación

Lenguaje de alto nivel

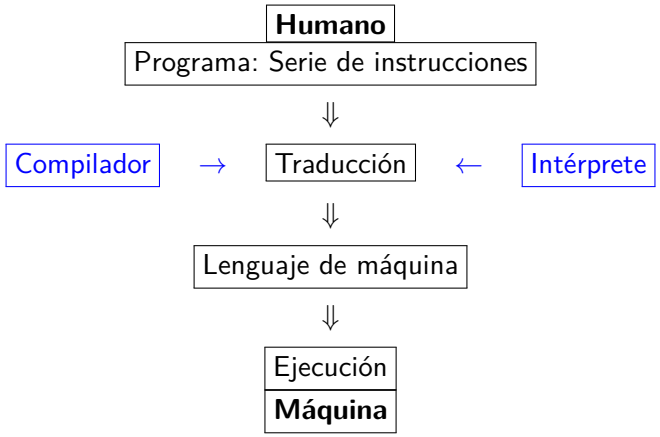
- Más cercanos a los lenguajes natural y matemático:

$$\text{resultado} = \text{dato1} + \text{dato2}$$

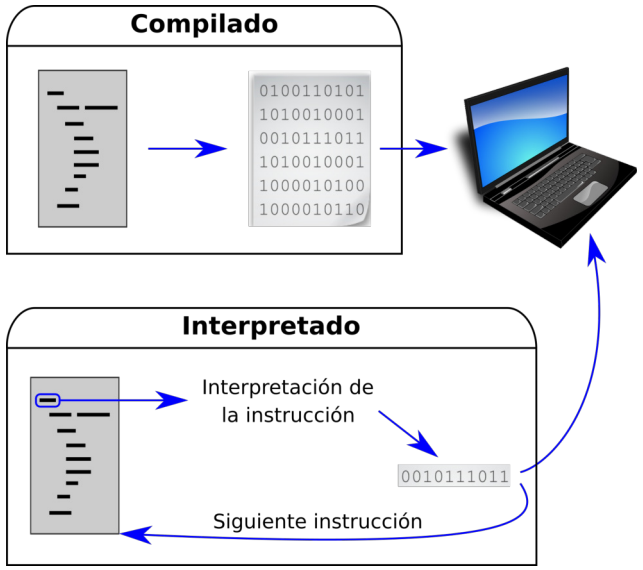
- Mayor legibilidad y facilidad de codificación.
- Estructuración de datos.
- Especializados por áreas.

Lenguajes de programación

Compilados vs. Interpretados (Alto nivel)



Lenguajes de programación



Programación estructurada

Programación estructurada

Es un estilo de programación basado en el teorema del programa estructurado (Corrado Böhm y Giuseppe Jacopini, 1966; Edsger Dijkstra, 1968), el cual establece que todo programa o algoritmo se puede desarrollar utilizando únicamente tres estructuras básicas de control:

- **Secuencia:** Ejecución sucesiva de instrucciones.
- **Selección:** Ejecución condicional de una o mas instrucciones, según el valor de una variable booleana (decisión).
- **Iteración:** Repetición de un conjunto de instrucciones mientras se cumple una condición (ciclo o bucle).

Programación estructurada

Características

- Control de ejecución de una línea de código a la siguiente.
- El programa se divide en segmentos más sencillos o de menor complejidad, llamados módulos o funciones.
- Principal diferencia con la programación clásica usada hasta hace unas décadas → No se utiliza la instrucción ***GOTO***.
- ***GOTO*** permite transferir el control de una parte del programa a otra, alterando la secuencia de ejecución.

Programación estructurada

Comparación

Se desea hacer un programa donde el usuario proporcione su edad, y el sistema imprima un mensaje que se sólo pueden leer personas mayores de 18 años, de modo que si el usuario ingresa su edad, y ésta es menor o igual a 18, entonces debe aparecer un mensaje de advertencia y terminar el programa. Si el usuario ingresa una edad mayor que 18, entonces el programa imprime el mensaje de interés y termina.

Programación con GOTO

Entradas:

edad: Almacena el valor de la edad de la persona

```
1 Inicio
2     Lea edad
3     Si edad <= 18 Entonces  GOTO 5
4     GOTO 7
5     Imprima 'No tiene acceso'
6     GOTO 9
7     Imprima 'Tiene acceso'
8     Imprima 'Puede leer el mensaje'
9 Fin
```

Programación estructurada

Entradas:

edad: Almacena el valor de la edad de la persona

```
1 Inicio
2     Lea edad
3     Si edad<=18 Entonces
4         Imprima 'No tiene acceso'
5     Sino
6         Imprima 'Tiene acceso'
7         Imprima 'Puede leer el mensaje'
8     Fin si
9 Fin
```


Programación estructurada

Ventajas

- Los programas se pueden leer en secuencia, desde el comienzo hasta el final, sin perder la continuidad de la tarea que cumplen.
- Los programas son más fáciles de entender por los usuarios.
- Se logra una reducción del esfuerzo en las pruebas de depuración del código (*debugging*).
- Se crean programas más sencillos y más rápidos.
- El esfuerzo de resolver varios subproblemas de forma aislada es con frecuencia menor que el de abordar el problema global.
- Se facilita el trabajo simultáneo en paralelo de distintos grupos de programadores.
- Posibilita en mayor grado la reutilización del código (especialmente de algunos módulos) en futuras aplicaciones.

Algoritmos

Mohammed al-KhoWârizmi → matemático persa (S. IX)

⇓

Reglas paso a paso para sumar, restar, multiplicar y dividir números decimales

Traducción al latín del apellido: *algorismus* → algoritmo.

Algoritmo

Conjunto de reglas finitas y precisas, ordenadas de forma lógica, que se ejecutan para desarrollar un cálculo, para realizar una tarea o actividad, para solucionar un problema, o para obtener una respuesta, ya sea de forma manual o automática.

Algoritmos

Ejemplo: algoritmo para cambiar una bombilla

1. **Inicio**
2. Colocar un escalera con acceso a la bombilla fundida
3. Elegir una bombilla con las características adecuadas
4. Subir por la escalera hasta alcanzar la bombilla fundida
5. Desmontar la bombilla fundida
6. Montar la bombilla nueva
7. Bajar de la escalera
8. Probar el funcionamiento
9. Retirar la escalera
10. **Fin**

Características

Un algoritmo debe cumplir con lo siguiente:

- **Ser preciso:**
  Debe poseer una serie de pasos definidos claramente y en orden lógico.
- **Estar bien definido:**
  Para un conjunto de datos idénticos, siempre se deben obtener los mismos resultados.
- **Ser finito:**
  Debe producir un resultado o salida en un determinado tiempo, es decir, debe tener una duración finta de ejecución (corto).

Clasificación

Clases de algoritmos

1. **Cualitativos:** No involucran cálculos numéricos.
2. **Cuantitativos:** Involucran cálculos numéricos.

Cualitativos

- Los pasos que realiza un estudiante para preparar un examen.
- Los pasos que se deben seguir para preparar una receta culinaria.

Cuantitativos

- Hallar el valor de la variable y, dado el valor de x, donde $y = x^2 + x - 3$
- Hallar el valor de la siguiente fórmula:
  $\texttt{Tot} = \texttt{SB} + \texttt{Hex} - \texttt{Desc}$

Solución de problemas mediante algoritmos

"Un algoritmo es un método para resolver un problema."

Problema → **Diseño del algoritmo** → **Programa**

"En la ciencia de la computación y en la programación, los algoritmos son más importantes que los lenguajes de programación o los computadores. Un lenguaje de programación es tan solo un medio para expresar un algoritmo y un computador es sólo un procesador para ejecutarlo."

Solución de problemas mediante algoritmos

1. Análisis del problema

- ¿Qué entradas se requieren? → Tipo y cantidad de datos
- ¿Qué salidas se requieren? → Cuáles resultados, de qué tipo, y qué cantidad.
- ¿Método adecuado para obtener las salidas deseadas?
- Consulta con expertos del área del conocimiento.
- Necesidades especiales del cliente/usuario (portabilidad, limitaciones físicas, etc.).

⇓

Qué se debe hacer

Solución de problemas mediante algoritmos

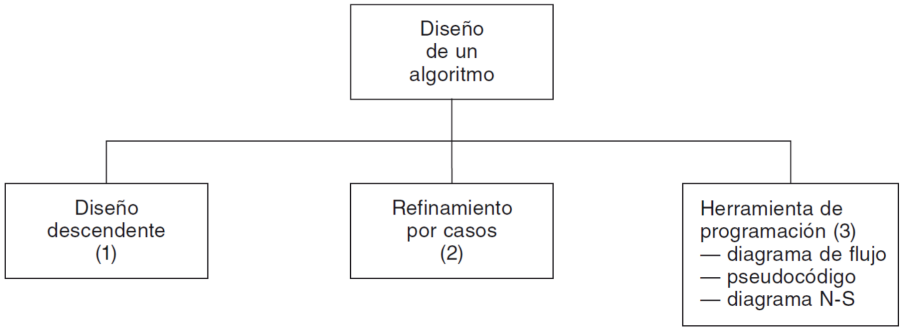
2. Diseño del algoritmo

- Definición de los pasos a seguir para la solución del problema.
- Primera descripción: Inicialmente pocos pasos (≤ 12).
- **Refinamiento del algoritmo**: Descripción más detallada con más pasos específicos.
- Problemas complejos: Varios niveles de refinamiento hasta obtener un algoritmo claro, preciso y completo.
- **Diseño descendente/modular**: División del problema en subproblemas de menor complejidad, los cuales también se pueden subdividir.
- Método de representación: diagramas de flujo o pseudocódigo.

⇓

Cómo se debe hacer

Solución de problemas mediante algoritmos



Solución de problemas mediante algoritmos

3. Prueba de escritorio

Ejecución manual del algoritmo (en papel) con el fin de verificar su adecuado funcionamiento e identificar posibles errores[a].

[a] Más adelante se dará una descripción detallada

4. Codificación

- Seleccionar un lenguaje de programación para codificar el algoritmo en un programa (Sintaxis y semántica).
- Debido a que el algoritmo es independiente del lenguaje de programación utilizado, es posible implementarlo fácilmente en cualquier lenguaje.

Solución de problemas mediante algoritmos

5. Ejecución y validación del programa

- Ejecución del programa para determinar si éste tiene errores (*"bugs"*).
- Amplia variedad de datos de entrada: valores normales, extremos y especiales, con los cuales sea posible comprobar todos los aspectos del programa.
- **Depuración** (*debugging*): Proceso de encontrar y corregir los errores de un programa.

6. Documentación y mantenimiento

- **Documentación interna**: Se incluye dentro del código del programa mediante comentarios que ayudan a su comprensión.
- **Documentación externa**: Manuales técnicos, manuales de usuario, anexos, etc.
- **Mantenimiento**: Actualizaciones y modificaciones del código. Grandes ($1.0 \rightarrow 2.0$), Menores ($1.0 \rightarrow 1.1$).

Tipos de errores

Errores de compilación:

- Uso incorrecto de las reglas del lenguaje de programación (errores de sintaxis).
- El computador no puede comprender la instrucción (No habrá ejecución, se imprimirá una lista de errores).

Errores de ejecución:

- Instrucciones que el computador puede comprender pero no ejecutar.
- Ejemplos: División por cero y raíces cuadradas de números negativos.
- Se detiene la ejecución del programa y se imprime un mensaje de error.

Tipos de errores

Errores lógicos:

- Errores en la lógica del programa $\rightarrow$ la fuente del error suele ser el diseño del algoritmo.
- Son los más difíciles de detectar: el programa puede ejecutarse y no producir errores de compilación ni de ejecución. Error $\rightarrow$ resultados incorrectos.
- Se debe volver a la fase de diseño, modificar el algoritmo, cambiar el programa fuente, y compilar y ejecutar una vez más.

$$\Downarrow$$

Importancia de la prueba de escritorio

A continuación

Próxima clase

Conceptos básicos:

- Variables
- Tipos de datos: Numéricos, alfanuméricos
- Operadores: Aritméticos, de relación, lógicos
- Representación de algoritmos: Diagramas de flujo.