PROGRAMACIÓN Y MÉTODOS NUMÉRICOS
2503506

CONCEPTOS BÁSICOS – 2

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

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Contenido

Pseudocódigo

Pseudocódigo

Es un lenguaje de especificación de algoritmos, basado en palabras y comandos.

- Su uso facilita significativamente el paso de codificación final.
- Nació como un lenguaje similar al inglés para representar las estructuras de control de programación estructurada. → Se ha traducido al español. → Subjetividad
- Se considera un primer borrador, dado que hace falta traducirlo posteriormente a un lenguaje de programación.
- No está hecho para ejecutarse en un computador.
  → PSeInt (`http://pseint.sourceforge.net/`)
- Normalmente requiere indentación (uso de sangrías) de diferentes líneas para hacer el algoritmo más legible.

Pseudocódigo

Instrucciones

Las instrucciones o comandos que se usan en pseudocódigo son palabras que indican acciones claras, comunes en programación.

Instrucciones comúnmente usadas

Instrucción	**Inglés**	**Español**
Comienzo de proceso	`begin`	`inicio`
Fin de proceso	`end`	`fin`
Entradas (lectura)	`read`	`leer`
Salidas (escritura)	`write`	`escribir`
Asignación	`A` ← 5	`A` ← 5

Pseudocódigo

Características

- Durante la planificación de un programa, el programador se puede concentrar en la lógica y en las estructuras de control y no preocuparse de las reglas de un lenguaje específico.
- Si se descubren errores o anomalías en la lógica del programa, es más fácil modificar ésta en el pseudocódigo que en el código de un lenguaje de programación.
- Se puede traducir fácilmente a lenguajes estructurados como Python, M, Pascal, C, FORTRAN, C++, Java, C#, etc.

Pseudocódigo – Indentación

```
Inicio
leer edad
si edad <= 18 entonces
escribir 'No tiene acceso'
sino
escribir 'Puede leer el mensaje'
fin si
Fin
```

```
Inicio
    leer edad
    si edad <= 18 entonces
        escribir 'No tiene acceso'
    sino
        escribir 'Puede leer el mensaje'
    fin si
Fin
```


Pseudocódigo

Estructura de un algoritmo en pseudocódigo

Consta de dos partes:

1. **Encabezado:**
   Comienza con la palabra **algoritmo**, y está seguida por el nombre que se da a éste.
2. **Cuerpo:**
   Es en sí el resto del algoritmo. Contiene dos secciones:
   - Instrucciones de declaración:
     En ésta se definen o declaran todas las variables utilizadas en el algoritmo → tipo y nombre
     - `Entradas, intermedias, salidas, var`.
   - Instrucciones ejecutables:
     Comprende las acciones que se deben ejecutar en el algoritmo.

Comentarios: Se pueden usar en ambas secciones para aclarar el código.

Pseudocódigo

Estructura de un algoritmo en pseudocódigo

```
algoritmo Nombre del algoritmo     % Encabezado[a]
% Sección de declaraciones
var tipo de datos: Lista de nombres de variables
inicio
    <Sentencia 1>
    <Sentencia 2>              % Cuerpo del algoritmo
        ⋮
    <Sentencia n>
fin
```

[a] Los comentarios en pseudocódigo se pueden indicar mediante el carácter %. En cada lenguaje de programación existe un carácter particular (# para Python, % para Matlab, etc.)

Pseudocódigo

Ejemplo: Cálculo del salario

```
algoritmo Salario
var   string:  nombre                    % Apellidos y primer nombre
        real:  horas, valor_h, imp, S_bruto, S_neto
inicio
    leer nombre, horas, valor_h
    S_bruto ← horas * valor_h            % Salario bruto
    imp ← 0.25 * S_bruto                 % Impuestos
    S_neto ← S_bruto − imp               % Salario neto
    escribir nombre, S_bruto, imp, S_neto
fin
```


Comparación de los métodos de representación

Diagramas de flujo

Ventajas:

- Facilidad de comprensión → Dibujo.
- Facilitan la comunicación entre programadores y clientes.
- Apto para problemas complejos → modularidad.

Desventajas:

- No permite mucho detalle en los pasos de solución.
- Puede ser poco práctico → ↑↑ Espacio.
- Construcción más lenta.
- Se complica para algoritmos extensos.
- Poca repetibilidad → Ramificaciones y conexiones.

Comparación de los métodos de representación

Pseudocódigo

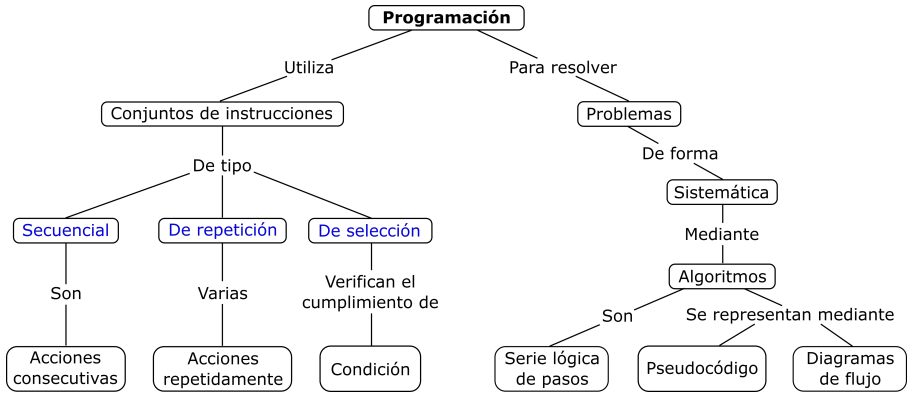
Ventajas:

- Construcción y verificación rápidas.
- Facilidad de modificación.
- Facilidad de traducción a código.
- Adaptado a la programación estructurada.
- Ocupa poco espacio.
- Al usar indentación se refleja la estructura del programa.

Desventajas:

- Requiere conocer ciertas palabras particulares.
- Puede llegar a ser muy detallado (Clientes).
- Puede ser extenso para problemas complejos.

Estructuras de programación

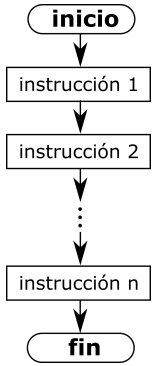


Estructura secuencial

Estructura secuencial

Consiste en establecer una secuencia de instrucciones (acciones).

- Pueden ser de lectura, asignación o escritura.
- Se escriben una a continuación de la otra, y se ejecutan en ese orden.



algoritmo Estructura secuencial

Entradas: Definición de las variables de entrada
Intermedias: Definición de variables intermedias
Salidas: Definición de las variables de salida

```
1 inicio
2     Instrucción 1
3     Instrucción 2
      ⋮
4     ⋮
5     Instrucción n
6 fin
```


Instrucciones de asignación

Se utilizan para asignar el valor de una variable.
La asignación es una operación destructiva, ya que reemplaza el valor anterior que pueda tener la variable, el cual se pierde.

Representación: `variable` ← Valor a asignar

Posibilidades de asignación

- Valor numérico:
  `T1` ← −28
- Valor de otra variable:
  `MEN` ← `R1`
- Expresión aritmética o algebraica:
  `Tot` ← `SB` + `HEX` − `D`
- Carácter o cadena de caracteres:
  `C` ← 'Programación'

Instrucciones de asignación

De las siguientes instrucciones de asignación determine cuáles son correctas, cuáles incorrectas, y por qué:

X	1. 1V ← 48	Nombre variable
✓	2. V_1 ← −48	
X	3. $R ← X + 2*F	Nombre variable
X	4. 27 ← TOT	Términos invertidos
✓	5. R ← 29	
✓	6. F1 ← M	M debe estar definida
X	7. 5 + X ← 12	Mal definido
X	8. SIL_ CAR ← 23000	Nombre variable
X	9. P − C ← F	Términos invertidos
✓	10. RES ← 0.04*(X + 27)	X debe estar definida
✓	11. TS ← (−X + 2)*(X − 2)	X debe estar definida

Prueba de escritorio

Prueba de escritorio

Consiste en ejecutar a 'mano' cada una de las instrucciones que componen el algoritmo → Simulación usando datos significativos.

- Se debe registrar cada una de las variables involucradas, con el fin de verificar que se cumple con las especificaciones deseadas.
- Se crea una tabla con una columna por cada variable, y al recorrer paso a paso el algoritmo, se registran los valores que van tomando las variables. → Una fila por cada paso de ejecución.
- Para indicar la escritura de variables (salida en pantalla), se crea una columna adicional denominada pantalla.

	Var_1	Var_2	···	Var_n	Pantalla
Paso 1					
Paso 2					
⋮					

Ejemplo 1 – Parámetros de un rectángulo

Problema

Elaborar un algoritmo que, dados su base y altura, permita calcular el perímetro y el área de un rectángulo, y posteriormente muestre los resultados.

Solución

1. **Análisis de la solución:**
   En primer lugar se deben conocer las fórmulas para calcular el área y el perímetro de un rectángulo:

$$\text{Área} = \text{Base} * \text{Altura}$$

$$\text{Perímetro} = 2 * (\text{Base} + \text{Altura})$$

Ejemplo 1 – Parámetros de un rectángulo

Solución

2. **Definición de variables:**
   - Entradas:
     Se requiere una variable con la cual leer la longitud de la base (`base`) y otra para leer el valor de la altura (`altura`).
   - Salidas:
     Se requiere una variable para almacenar el valor del área (`area`) y otra para el valor del perímetro (`perimetro`).

   En este caso no hay necesidad de variables intermedias.

Ejemplo 1

3. **Pseudocódigo:**

algoritmo Parámetros de un rectángulo

Entradas:

```
real:   base        % Base del rectángulo [cm]
        altura      % Altura del rectángulo [cm]
```

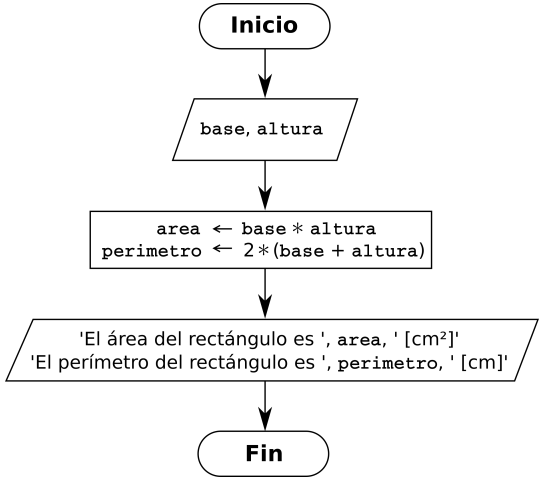
Salidas:

```
real:   area        % Área del rectángulo [cm^2]
        perimetro   % Perímetro del rectángulo[cm]
```

```
1 inicio
2     leer base, altura
3     area ← base * altura
4     perimetro ← 2 * (base + altura)
5     escribir 'El área del rectángulo es ', area, ' [cm^2]'
6     escribir 'El perímetro del rectángulo es ', perimetro, ' [cm]'
7 fin
```


Ejemplo 1

4. **Diagrama de flujo:**



Ejemplo 1

5. **Prueba de escritorio:** 5 pasos, `base` = 4 cm, `altura` = 8 cm

	`base`	`altura`	`area`	`perimetro`	Pantalla
Paso 1					
Paso 2					
Paso 3					
Paso 4					
Paso 5					

Ejemplo 1

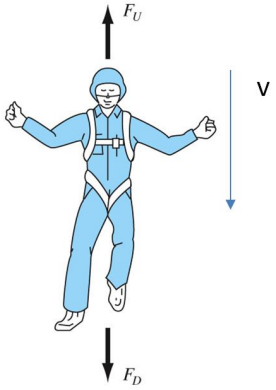
5. **Prueba de escritorio:** 5 pasos, `base` = 4 cm, `altura` = 8 cm

	`base`	`altura`	`area`	`perimetro`	Pantalla
Paso 1	4	8			
Paso 2	4	8	32		
Paso 3	4	8	32	24	
Paso 4	4	8	32	24	El área del rectángulo es 32 [cm^2]
Paso 5	4	8	32	24	El perímetro del rectángulo es 24 [cm]

Ejemplo 2 – Velocidad de caída de un paracaidista

Planteamiento

Realice un algoritmo para determinar la velocidad de caída de un paracaidista en cualquier instante. Éste salta con velocidad vertical inicial igual a cero, y está sujeto a la fuerza gravitacional (F_D) y la fuerza de arrastre aerodinámico (F_U), la cual es proporcional a su velocidad de caída.



Ejemplo 2

Análisis

El paracaidista está en condiciones de caída libre, de modo que se puede emplear la segunda ley de Newton para describir el fenómeno:

$$m\frac{dV(t)}{dt} = F_D + F_U \qquad \text{Segunda ley de Newton}$$

$$F_D = mg \qquad \text{Fuerza gravitacional}$$

$$F_U = -CV(t) \qquad \text{Fuerza de arrastre aerodinámico}$$

Donde:

- t : Tiempo [s]
- $V(t)$: Velocidad de caída [m/s]
- F_D : Fuerza debida a la aceleración gravitacional [N]
- F_U : Fuerza debida al arrastre [N]
- m : Masa del paracaidista [kg] (cte.)
- C : Coeficiente de arrastre [Ns/m]

Ejemplo 2

Análisis

El modelo que describe el fenómeno es el siguiente:

$$m\frac{dV(t)}{dt} = mg - CV(t) \quad \Rightarrow \quad \boxed{\frac{dV(t)}{dt} + \frac{C}{m}V(t) = g}$$

Se tiene una ecuación diferencial ordinaria de primer orden, no homogénea, con coeficientes constantes.

Solución analítica: Método del factor integrante.

$$\boxed{V(t) = \frac{mg}{C}\left[1 - \exp^{-(\frac{C}{m})t}\right]}$$

Ejemplo 2

Diseño del algoritmo

Entradas:

- Masa del paracaidista: `m`
- Coeficiente de arrastre: `C`
- Tiempo de caída: `t`

Variables intermedias:

- Aceleración de la gravedad: `g`

Salidas:

- Tiempo de caída: `t`
- Velocidad de caída: `V`

Ejemplo 2

Diseño del algoritmo

Pasos:

1. Leer los datos de entrada (`m`, `C`, `t`)
2. Crear constante con el valor de la aceleración gravitacional: `g`
3. Calcular la velocidad de caída correspondiente al tiempo t: `V`
4. Escribir los resultados: Velocidad de caída, $V(t)$, en el tiempo dado (t): `V`, `t`

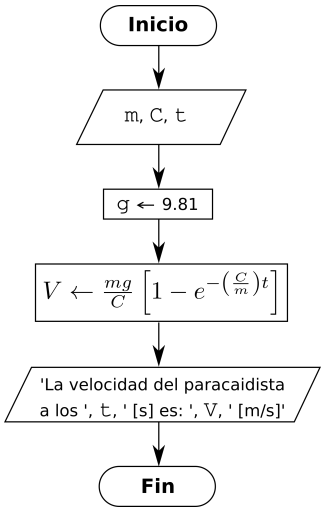
Ejemplo 2

algoritmo Velocidad de caída de un paracaidista

Entradas:	real:	`m`	`%Masa del paracaidista [kg]`
		`C`	`%Coeficiente de arrastre [Ns/m]`
		`t`	`%Tiempo de caída [s]`
Intermedias:	real:	`g`	`%Aceleración gravitacional [m/s^2]`
Salidas:	real:	`V`	`%Velocidad de caída del paracaidista [m/s]`
		`t`	

```
1 inicio
2     leer m, C, t
3     g ← 9.81                                          %[m/s^2]
4     V ← mg/C [1 − e^−(C/m)t]                          %Velocidad de caída [m/s]
5     escribir 'La velocidad del paracaidista a los ', t ' [s] es: ', V, ' [m/s]'
6 fin
```

Ejemplo 2



Ejemplo 2

Prueba de escritorio: 4 pasos

m = 68.1 kg, C = 12.5 Ns/m, t = 10 s, g = 9.81 m/s^2

	m	C	t	g	V	Pantalla
Paso 1						
Paso 2						
Paso 3						
Paso 4						

Ejemplo 2

Prueba de escritorio: 4 pasos

m = 68.1 kg, C = 12.5 Ns/m, t = 10 s, g = 9.81 m/s^2

	m	C	t	g	V	Pantalla
P. 1	68.1	12.5	10			
P. 2	68.1	12.5	10	9.81		
P. 3	68.1	12.5	10	9.81	44.92	
P. 4	68.1	12.5	10	9.81	44.92	La velocidad del paracaidista a los 10 [s] es: 44.92 [m/s]

Ejercicios

Simples: Página web del curso

- Asignación.
- Identificación del objetivo de un algoritmo.
- Saludo personalizado.
- Intercambio de valores de variables.
- Índice de masa corporal.

A continuación

Próxima clase

Presentación de Python:

- Presentación.
- Entornos de trabajo (IDE).
- Operaciones matemáticas básicas.
- Ejemplos.