PROGRAMACIÓN Y MÉTODOS NUMÉRICOS
2503506

CONCEPTOS BÁSICOS – 1

Andrés Agudelo
Departamento de Ingeniería Mecánica

UNIVERSIDAD DE ANTIOQUIA
Facultad de Ingeniería

Contenido

Variables

Definición

Las variables son espacios reservados en la memoria principal que, como su nombre indica, pueden cambiar de valor a lo largo de la ejecución de un programa.

Están compuestas por dos partes: nombre y valor o contenido.

Por ejemplo: Costo = 57505

Nombre

Conjunto de caracteres (letras, números y caracteres especiales aceptados) con los cuales se identifica el contenido del espacio en memoria en todo momento.

Valor

Son los datos que una variable representa o tiene asociados en un momento determinado.

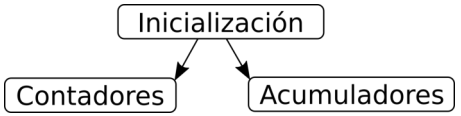
Variables

Inicialización

En algunas ocasiones, el programador establece un valor inicial para algunas variables, de modo que éste será su valor cuando se usen por primera vez.

Dicho valor inicial puede cambiar durante la ejecución del programa.

Esta práctica se conoce como inicializar una variable, y es necesaria en la solución numérica de muchos problemas matemáticos.



Creación de variables

Reglas

- **El nombre debe comenzar con una letra.** La mayoría de lenguajes de programación no permiten iniciar el nombre de la variable con un número o símbolo.

  `T` `Precio` `34Sur` `_tiempo`

- El nombre debe ser nemotécnico, es decir, debe estar relacionado con su contenido. → Muy importante para tener algoritmos claros y fáciles de depurar.

  Temperatura → `T, Temp` Velocidad → `V, Vel`

Creación de variables

Reglas

- Existe un límite para la extensión del nombre de las variables (depende del lenguaje de programación) → Descriptores cortos.

  `Buril` `Herramientadecorteparatorno`

- **No se permiten espacios** entre las letras del nombre.

  `Tagua, Taceite, Vcorte`
  `presion vapor, momento inercia`

- **No se debe usar caracteres especiales**, excepto los permitidos, como el guión inferior (_).

  `T_agua, T_aceite, V_corte`

Creación de variables

Reglas

- Los nombres de las variables no deben coincidir con palabras reservadas en un lenguaje de programación (comandos o constantes).

  `altura, tiempo, posicion`
  `sin, cos, print, import, sqrt`

- Se debe distinguir entre letras mayúsculas y minúsculas, dependiendo del lenguaje de programación.

  `Presion` $\neq$ `presion`

Ejemplos

`TOT`	`CODIGO`	`codigo`	`COD`	`Cod`
`men100f`	`nombre`	`Nombre`	`NOMBRE`	`NOM`
`Suel_X`	`Costo005`	`Suel1_T`	`Nivel_agua`	`Nivel_aceite`

Asignación

Asignación

Consiste en darle valor a una variable, bien sea al inicializarla, o mediante un proceso durante la ejecución del algoritmo (por ejemplo, el resultado de un cálculo).

El operador que se usa para indicar la asignación en la representación de algoritmos es una flecha horizontal hacia la izquierda:

nombre de la variable ← valor

Ejemplos

```
vel_ini ← 17.35
potencia_eje ← par * vel_angular
peso_total ← 2 * peso_motor + peso_estructura
```

Asignación

Asignación

- Las variables no tienen restricciones en cuanto a su contenido, es decir, pueden almacenar cualquier tipo de datos.
- Si el contenido de una variable es un carácter o una cadena de caracteres, éste debe asignarse entre comillas simples (' ').
- En algunos lenguajes de programación existen tipos de variables inmutables.

Ejemplos

```
TOT ← 32456          Sueldo ← 1553000
S1_X ← −875.5        Men25 ← 'Bienvenidos'
C ← '$'              R ← '*'
```


Constantes

Constantes

Nombre que se da a un campo de memoria cuyo contenido no cambia a lo largo de un proceso.

Ejemplos:

```
pi ← 3.1416      R1X ← −34.785      g ← 9.81
```

El nombre de las constantes se asigna teniendo en cuenta las mismas normas que se usan para la definición de variables.

Tipos de datos

Numéricos

Se dividen en enteros y reales:

- **Enteros (*integer*, int)**
  Son números que pueden estar precedidos de los signos + ó −, y no tienen parte decimal.

  Ejemplo: −24 35

  ⇒ Contadores, indicadores de posición (≥ 0).

- **Reales o de coma flotante (float)**
  Son números que pueden estar precedidos de los signos + ó −, y tienen tanto parte entera como decimal.

  Ejemplo: −25.8 85.32

Tipos de datos

Alfanuméricos

Pueden contener un carácter o una cadena de caracteres: ' '

- **Carácter (*character*)**
  Es cualquier letra (**A – Z** o **a – z**), número (**0 – 9**) o símbolo (#,$, %, –, etc.).
  Ejemplo: 'a', '8', '%', '–'.
- **Cadena de caracteres (*string*, str)**
  Es una sucesión de caracteres.
  Ejemplo: 'Hola mundo', 'Universidad', 'Calle 66 #32–25', 'a*b', '58634'

Datos lógicos o booleanos (*bool*)

Son datos que sólo pueden tomar dos valores: verdadero (**True**) o falso (**False**). Por ejemplo, dado un valor de A y de B, determinar el valor de la relación lógica $A > B$.

Operadores

Operador

- Símbolo o palabra que permite formular operaciones, establecer relaciones o hacer comparaciones de tipo lógico y matemático.

Existen tres tipos principales de operadores:

1. Aritméticos
2. De relación
3. Lógicos

Operadores aritméticos

Operador	Descripción
$+$	Suma
$-$	Resta
$*$	Multiplicación
$/$	División
$**$, $\wedge$	Potenciación
$//$[1]	División entera
$\%$[2]	Módulo o residuo

[1] Se utiliza sólo en algunos lenguajes de programación
[2] Calcula el residuo de la división entre dos números

Jerarquía de operaciones aritméticas

Jerarquía de operaciones

La siguiente jerarquía se sigue en la mayoría de lenguajes de programación, incluido Python:

1. Potenciación.
2. Multiplicación y división.
3. Suma y resta.

Otras consideraciones

- Si los operadores están en el mismo nivel de jerarquía, las operaciones se realizan en orden de aparición: de izquierda a derecha.
- Si una instrucción contiene paréntesis, se realizan en primer lugar las operaciones que se encuentran en el paréntesis más interno, siguiendo el nivel de jerarquía mencionado anteriormente.

Operadores de relación

Se utilizan para crear expresiones lógicas que dan como resultado una variable *booleana* (`verdadero, falso`).

Operador	Descripción
$>$	Mayor que
$<$	Menor que
$==$	Igual que
$>=$	Mayor o igual que
$<=$	Menor o igual que
$!=$ $(<>)$	Diferente

A $!=$ B indica que A es diferente de B:
$\rightarrow$ Si A es igual a B, la proposición será falsa
$\rightarrow$ Si A es diferente a B, la proposición será verdadera

Operadores lógicos

Se utilizan para comparar expresiones *booleanas* (valores lógicos de verdadero y falso), o para cambiar su valor.

Conjunción: **y** ⇒ **and** (&&)

Es verdadera sí y solo sí las expresiones comparadas son verdaderas, de lo contrario es falsa.

$(A < B)$	**y**	$(C > D)$	**Resultado**
Verdadero		Verdadero	Verdadero
Verdadero		Falso	Falso
Falso		Verdadero	Falso
Falso		Falso	Falso

Evaluación sucesiva condicional:

Verdadero → **y** →

Falso -X-> **y**

Operadores lógicos

Disyunción: **o** ⇒ **or** (||)

Es verdadera si se cumple al menos una de las relaciones, de lo contrario es falsa.

$(A < B)$	**o**	$(C > D)$	**Resultado**
Verdadero		Verdadero	Verdadero
Verdadero		Falso	Verdadero
Falso		Verdadero	Verdadero
Falso		Falso	Falso

Evaluación completa:

Verdadero → **o** →

Falso → **o** →

Operadores lógicos

Negación: **no** ⇒ **not** (∼)

Se usa para indicar negación ⇒ Cambia el valor de una variable *booleana*

$(A < B)$	**no** $(A < B)$
Verdadero	Falso
Falso	Verdadero

Cadenas de expresiones

Exp. 1 **y** Exp. 2 **y** ⋯ **y** Exp. n → Resultado

Exp. 1 **o** Exp. 2 **o** ⋯ **o** Exp. n → Resultado

Exp. 1 **y** Exp. 2 **o** Exp. 3 ⋯ **y no** Exp. n → Resultado

no (Exp. 1 **y no** Exp. 2 **o** Exp. 3 ⋯ **y** / **o** Exp. n) → Resultado

Representación de algoritmos

Representación

El conjunto de pasos para resolver un problema determinado se puede representar gráficamente o mediante texto.

- **Representación gráfica:**
  - Diagramas de flujo → Formas geométricas particulares y conectores.
  - Diagramas NS (Nassi-Schneiderman) → Un bloque con divisiones especiales.
- **Texto:**
  Lista secuencial de pasos detallados, usando un lenguaje particular, denominado pseudocódigo.

Representación de algoritmos

Diagrama de flujo

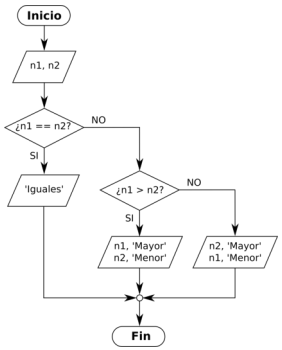
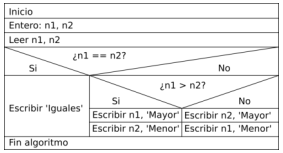


Diagrama NS



Pseudocódigo

algoritmo Comparar números

Entradas: Entero: n1, n2

```
1  Inicio
2      leer n1, n2
3      si n1 == n2 entonces
4          escribir 'Iguales'
5      sino
6          si n1 > n2 entonces
7              escribir n1,
                   'Mayor'
8              escribir n2,
                   'Menor'
9          sino
10             escribir n2,
                   'Mayor'
11             escribir n1,
                   'Menor'
12         fin si
13     fin si
14 Fin
```

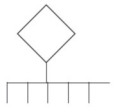

Diagramas de flujo

Diagramas de flujo (*Flowchart*)

Representación gráfica de un algoritmo.

- Es una de las técnicas de representación de algoritmos más antigua, aunque su uso ha disminuido considerablemente (lenguajes de programación estructurados).
- Muestra de forma gráfica la serie de pasos ordenados y lógicos que llevan a la solución de un problema.
- Es un diagrama que utiliza símbolos estándar (ANSI). → Cajas.
- Los pasos de un algoritmo se escriben en cajas unidas por flechas (líneas de flujo), que indican la secuencia en que se debe ejecutar.

Diagramas de flujo

Símbolos principales	Función
	Terminal (representa el comienzo, "inicio", y el final, "fin" de un programa. Puede representar también una parada o interrupción programada que sea necesario realizar en un programa.
	Entrada/Salida (cualquier tipo de introducción de datos en la memoria desde los periféricos, "entrada", o registro de la información procesada en un periférico, "salida".
	Proceso (cualquier tipo de operación que pueda originar cambio de valor, formato o posición de la información almacenada en memoria, operaciones aritméticas, de transferencia, etc.).
	Decisión (indica operaciones lógicas o de comparación entre datos —normalmente dos— y en función del resultado de la misma determina cuál de los distintos caminos alternativos del programa se debe seguir; normalmente tiene dos salidas —respuestas SÍ o NO— pero puede tener tres o más, según los casos).
	Decisión múltiple (en función del resultado de la comparación se seguirá uno de los diferentes caminos de acuerdo con dicho resultado).
	Conector (sirve para enlazar dos partes cualesquiera de un ordinograma a través de un conector en la salida y otro conector en la entrada. Se refiere a la conexión en la misma página del diagrama.

Diagramas de flujo

Símbolos principales	Función
	Indicador de dirección o línea de flujo (indica el sentido de ejecución de las operaciones).
	Línea conectora (sirve de unión entre dos símbolos).
	Conector (conexión entre dos puntos del organigrama situado en páginas diferentes).
	Llamada a subrutina o a un proceso predeterminado (una subrutina es un módulo independientemente del programa principal, que recibe una entrada procedente de dicho programa, realiza una tarea determinada y regresa, al terminar, al programa principal).
	Pantalla (se utiliza en ocasiones en lugar del símbolo de E/S).
	Impresora (se utiliza en ocasiones en lugar del símbolo de E/S).
	Teclado (se utiliza en ocasiones en lugar del símbolo de E/S).
	Comentarios (se utiliza para añadir comentarios clasificadores a otros símbolos del diagrama de flujo. Se pueden dibujar a cualquier lado del símbolo).

Diagramas de flujo

Normas de construcción

- Todo diagrama de flujo debe tener un inicio y un fin.
- Se debe usar líneas rectas verticales y horizontales para indicar la dirección de flujo (no diagonales).
- Todas las líneas utilizadas para indicar la dirección de flujo deben estar conectadas.
- El diagrama se debe elaborar de tal forma que siga visualmente el flujo de arriba hacia abajo y de izquierda a derecha.
- La notación utilizada en el diagrama debe ser independiente de un lenguaje de programación.

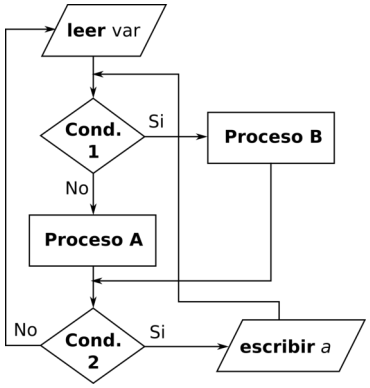
Diagramas de flujo

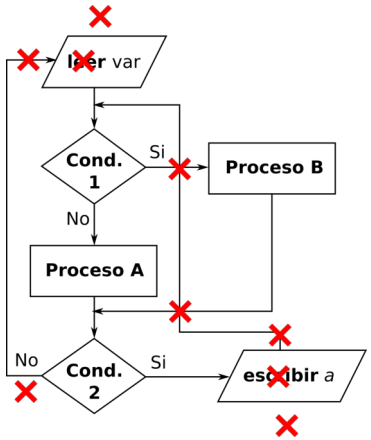
Normas de construcción

- Si el diagrama requiere de más de una hoja para su construcción, se deben usar los conectores apropiados para darle continuidad, y se debe enumerar cada página.
- No debe salir más de una línea de un mismo símbolo, excepto del bloque de decisión. No obstante, pueden llegar varias líneas de flujo a otras líneas.
- Las líneas de flujo no se deben cruzar. → Utilizar los conectores de flujo apropiados.
- Las líneas de flujo deben entrar a un símbolo por la parte superior y/o izquierda, y salir de éste por la parte inferior y/o derecha.

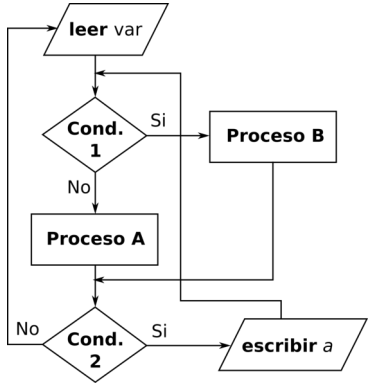
Diagramas de flujo

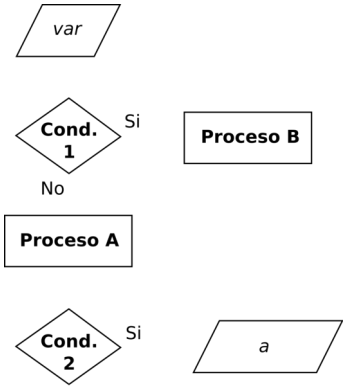
¿Qué problemas se identifican en el siguiente diagrama de flujo?



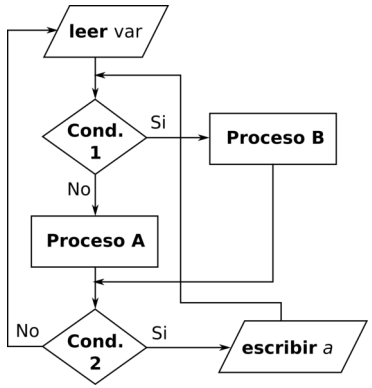


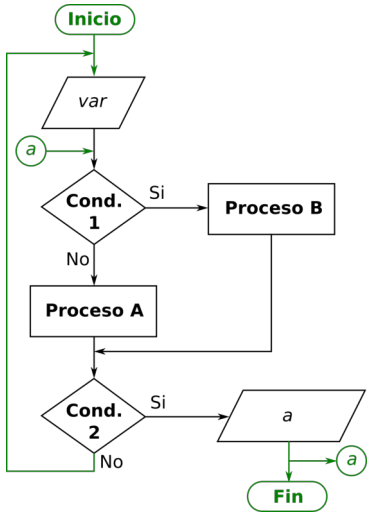
Diagramas de flujo





Diagramas de flujo





Diagramas de flujo

Ejemplo 1

Se debe calcular el salario bruto y el salario neto de un trabajador "por horas", conociendo su nombre, el número de horas trabajadas en la semana, y los impuestos a pagar (25 %).

Ejemplo 1 – Planteamiento

Información necesaria: (Entradas)

- Nombre del trabajador, Número de horas trabajadas en la semana, Valor de la hora.

Cálculos: (Procesos)

- Salario bruto, impuestos, salario neto.

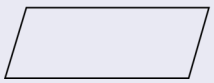
Resultados: (Salidas)

- Nombre del trabajador, salario bruto, salario neto.

Diagramas de flujo

Ejemplo 1 – Elementos del diagrama

Entrada / Salida



Procesos

Flujo de información

Diagramas de flujo

Ejemplo 1 – Definición de variables

Entradas:

- Nombre del trabajador → `nombre`
- Número de horas trabajadas en la semana → `horas`
- Valor de la hora → `valor_h`

Salidas:

- Nombre del trabajador → `nombre`
- Salario bruto → `S_bruto`
- Salario neto → `S_neto`

Intermedias:

- Impuestos → `imp`

Diagramas de flujo

Ejemplo 1 – Pasos

1. Leer la información de entrada:

   $$\texttt{nombre}, \texttt{horas}, \texttt{valor\_h}$$

2. Calcular el salario bruto:

   $$\texttt{S\_bruto} \leftarrow \texttt{horas} * \texttt{valor\_h}$$

3. Calcular los impuestos:

   $$\texttt{imp} \leftarrow \texttt{S\_bruto} * 0.25$$

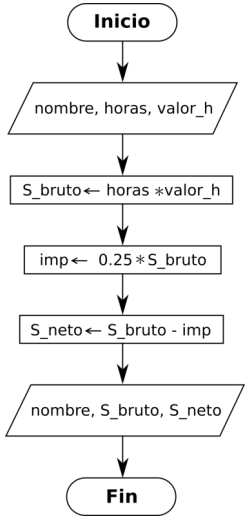
4. Calcular el salario neto:

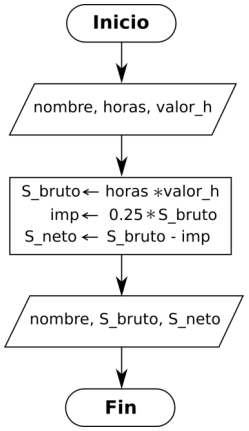
   $$\texttt{S\_neto} \leftarrow \texttt{S\_bruto} - \texttt{imp}$$

5. Mostrar resultados:

   $$\texttt{nombre}, \texttt{S\_bruto}, \texttt{S\_neto}.$$

Diagramas de flujo





Diagramas de flujo

Ejemplo 2

Determinar los salarios bruto y neto semanal de los empleados de una empresa, sabiendo que éstos se calculan con base en las horas semanales trabajadas, y de acuerdo a un valor especificado por hora ordinaria.

Información específica:

Si se superan cuarenta horas semanales, las horas extraordinarias se pagarán a razón de 1.5 veces el valor de la hora ordinaria.

Los impuestos para horas ordinarias son del 25 %, y para las extraordinarias son del 10 %.

Diagramas de flujo

Ejemplo 2 – Planteamiento

Información necesaria: (Entradas)

- Nombre del trabajador (`nombre`), Número de horas trabajadas en la semana (`horas`), valor de la hora ordinaria (`valor_ho`).

Cálculos: (Procesos)

- Salario ordinario (`S_ord`), salario extraordinario (en caso de que exista, `S_ext`), impuestos (`imp`), Salario bruto (`S_bruto`), salario neto (`S_neto`).

Resultados: (Salidas)

- Nombre del trabajador (`nombre`), Salario bruto (`S_bruto`), Salario neto (`S_neto`).

¿Variables intermedias?

Diagramas de flujo

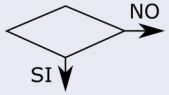
Ejemplo 2 – Elementos del diagrama

Entrada / Salida

Procesos

Flujo de información

Decisión



Diagramas de flujo

Ejemplo 2 – Pasos

1. Leer la información de entrada (`nombre`, `horas`, `valor_ho`)
2. Determinar si el número de horas trabajadas supera las 40/semana:
   - **Falso** ⇒ Todas las horas son ordinarias:

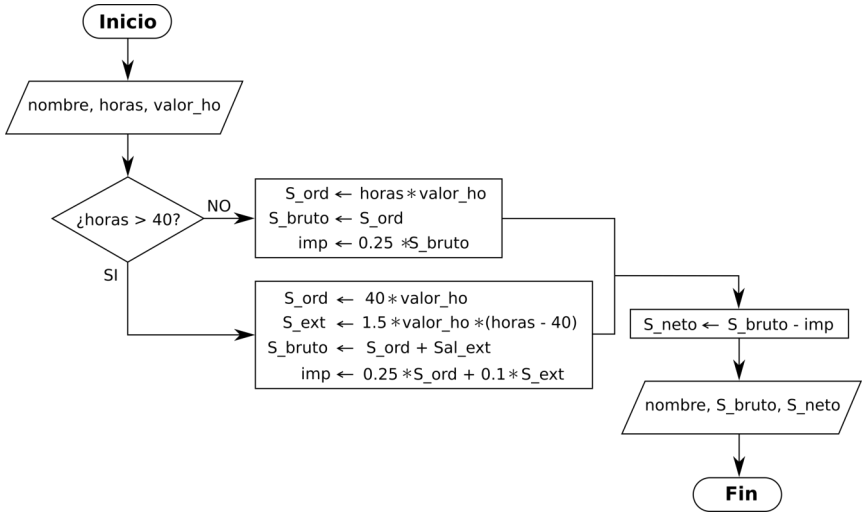
$$\begin{aligned} \texttt{S\_ord} &\leftarrow \texttt{horas} * \texttt{valor\_ho} \\ \texttt{S\_bruto} &\leftarrow \texttt{S\_ord} \\ \texttt{imp} &\leftarrow 0.25 * \texttt{S\_bruto} \end{aligned}$$

   - **Verdadero** ⇒ Se deben pagar horas extraordinarias:

$$\begin{aligned} \texttt{S\_ord} &\leftarrow 40 * \texttt{valor\_ho} \\ \texttt{S\_ext} &\leftarrow 1.5 * \texttt{valor\_ho} * (\texttt{horas} - 40) \\ \texttt{S\_bruto} &\leftarrow \texttt{S\_ord} + \texttt{S\_ext} \\ \texttt{imp} &\leftarrow 0.25 * \texttt{S\_ord} + 0.1 * \texttt{S\_ext} \end{aligned}$$

3. Calcular el salario neto: $\texttt{S\_neto} \leftarrow \texttt{S\_bruto} - \texttt{imp}$
4. Mostrar resultados: `nombre`, `S_bruto`, `S_neto`.

Diagramas de flujo – Ejemplo 2



A continuación

Próxima clase

- Pseudocódigo.
- Estructuras de programación: secuencial, asignación.
- Verificación de algoritmos: prueba de escritorio.