

Metodología de la Programación

Programación Estructurada y Modular

Programación estructurada:

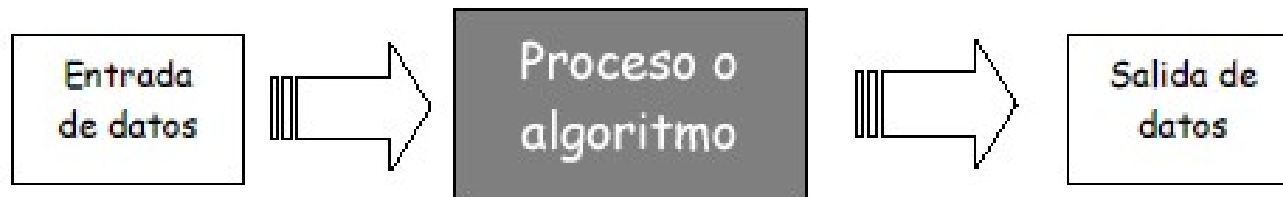
- Diseño descendente (top-down)
- Descomponer una acción compuesta en otras más simples.
- Uso de estructuras básicas de control (secuencial, alternativa, repetitiva)

Programación Modular: división o subdivisión de un programa en módulos programados y compilados en forma independiente de manera que cada uno de ellos tenga encomendada la ejecución de una única tarea o actividad.

Metodología de la Programación

Estructura general de un programa

- Entrada de datos
- Proceso o algoritmo
- Salida de datos o resultados



Metodología de la Programación

Representación de los algoritmos - Características

- *Independiente* del lenguaje de programación
- *Diseño normalizado*
- *Intuitivo*
- *Flexible*
- *Preciso*: no ambiguo (orden, contenido)
- *Determinístico*: mismos valores=mismo resultado
- *Finito*
- *General*: debe servir para una clase de problemas lo más amplia posible
- *Eficiente*: lo bueno si breve...

Metodología de la Programación

Diagramas de flujo de programas

❖ INICIO

❖ Secuencia de operaciones ordenada y detallada (arriba-abajo e izquierda-derecha)

❖ FIN

❖ Símbolos conectados por medio de líneas de flujo

❖ Las líneas de conexión no pueden cruzarse

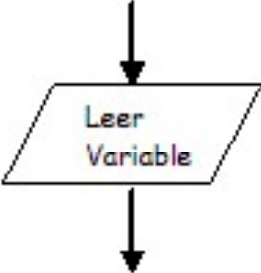
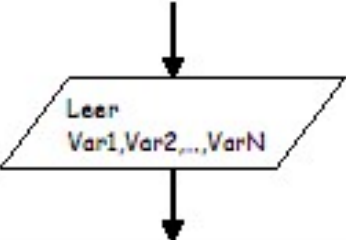
❖ Diagrama prolijo y claro

❖ Dibujarlo cuantas veces sea necesario hasta que "se vea lindo"

Metodología de la Programación

Instrucciones de entrada

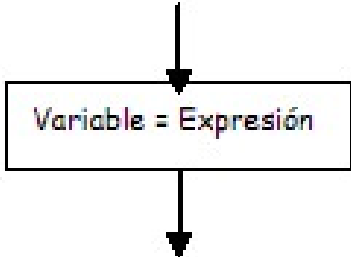
Encargadas de recoger datos de un dispositivo de entrada.

Representación en Ordinograma	Representación en Pseudocódigo
	Leer Variable
	Leer Var1, Var2, ... , VarN

Metodología de la Programación

Instrucciones de asignación

Encargadas de almacenar un dato obtenido al evaluar una expresión en una variable simple previamente declarada.

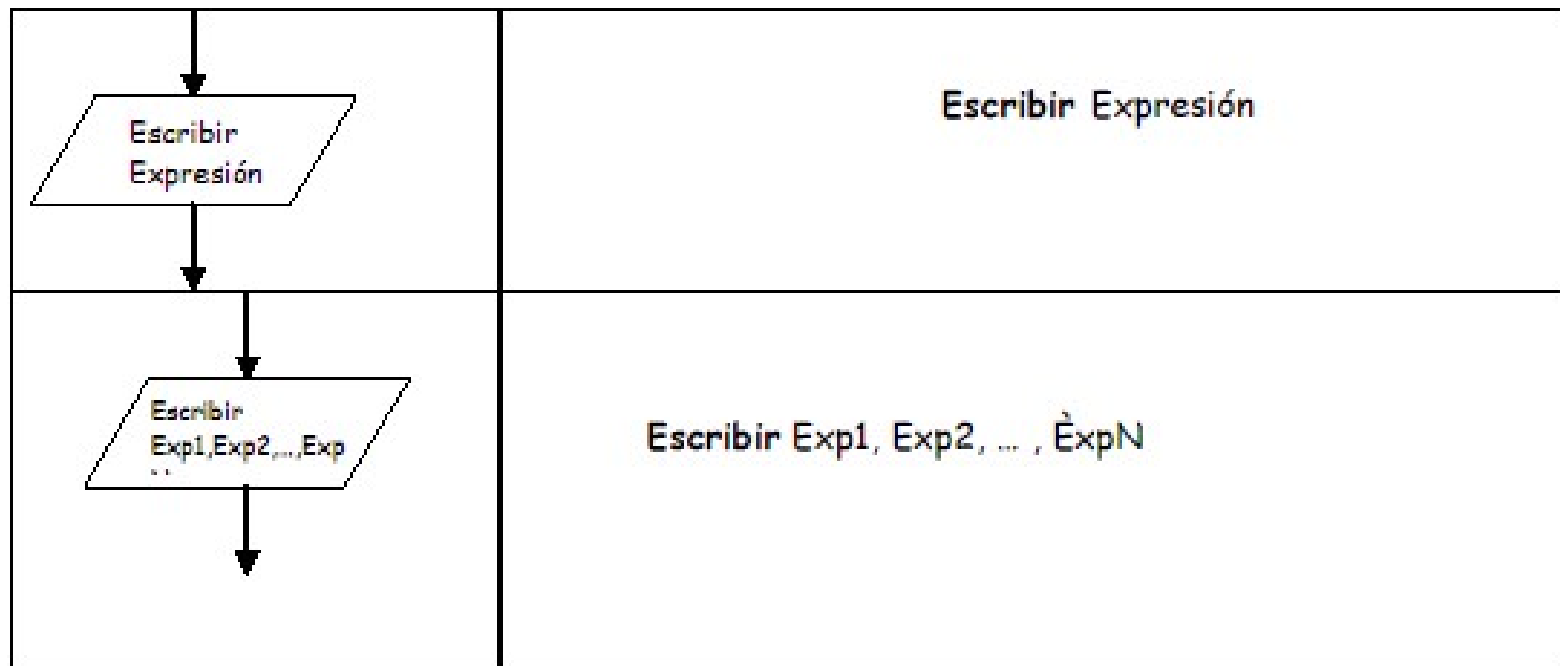
Representación en Ordinograma	Representación en Pseudocódigo
	<p>Variable = Expresión</p> <p>O bien:</p> <p>Variable ← Expresión</p>

- El tipo de variable en ambos lados de la asignación debe ser el mismo.
- Si asigno una expresión compleja, ésta será previamente evaluada.
- Dar un valor inicial (inicializar) a la variable antes de usarla.

Metodología de la Programación

Instrucciones de salida

Encargadas de enviar los datos de una variable o expresión a un dispositivo de salida.



Metodología de la Programación

Pseudocódigo

Representación no gráfica de un algoritmo.

Ventajas:

- ✓ Concentración en la lógica y estructuras de control del algoritmo y no en lenguaje de programación.
- ✓ Más fácil de crear y entender.
- ✓ Totalmente independiente del lenguaje de programación.
- ✓ Facilita futuras correcciones o actualizaciones.

Metodología de la Programación

Pseudocódigo

Reglas:

- ❖ INICIO y FIN
- ❖ Una instrucción por línea
- ❖ Palabras reservadas: si, entonces, para, mientras, etc.
- ❖ Usar indentación (tabulado) para mostrar dependencias de control
- ❖ Escrito en minúsculas excepto nombres de variable, módulos, etc.
- ❖ Partes de un programa en pseudocódigo: cabecera y cuerpo.

Metodología de la Programación

Pseudocódigo

Cabecera: bloque informativo

- nombre del programa, qué hace y nombre del/los autor/es.
- Si es una sola línea: `//`, sino `/* ... */`

Cuerpo: resto del diseño

- Bloque de datos: definición de variables
- bloque de acciones: descripción detallada de las órdenes y acciones a ejecutar.

Metodología de la Programación

Estructuras de control: alternativas o de decisión

Categoría de Instrucción	Representación en un ordinograma	Representación en un pseudocódigo
Alternativa Simple	<pre> graph TD Start(()) --> Cond{CONDICIÓN} Cond -- NO --> Join(()) Cond -- SI --> I[I] I --> Join Join --> End(()) </pre>	<p><i>Si CONDICIÓN entonces</i> <i>I1;</i> <i>I2; ... ;</i> <i>In;</i> <i>finsi</i></p>
Alternativa Doble	<pre> graph TD Start(()) --> Cond{CONDICIÓN} Cond -- NO --> I1[I] Cond -- SI --> I2[I] I1 --> Join(()) I2 --> Join Join --> End(()) </pre>	<p><i>Si CONDICIÓN entonces</i> <i>I1;</i> <i>I2; ... ;</i> <i>In;</i> <i>Si No</i> <i>J1;</i> <i>J2; ... ;</i> <i>Jn;</i> <i>finsi</i></p>
Alternativa Múltiple	<pre> graph TD Start(()) --> Cond{CONDICIÓN} Cond --> I1[I1] Cond --> IN[IN] Cond --> INplus1[IN+1] </pre>	<p><i>opción EXPRESIÓN de</i> <i>V1 hacer I1; I2; ... ; Im;</i> <i>V2 hacer J1; J2; ... ; Jn;</i> <i>...</i> <i>Vn hacer K1; K2; ... ; Ko;</i> <i>otro hacer L1; L2; ... ; Lp;</i> <i>fin_opcion</i></p>

Metodología de la Programación

Estructuras de control: repetitivas

Categoría de Instrucción	Representación en un ordinograma	Representación en un pseudocódigo
Instrucción Mientras:	<pre> graph TD Start(()) --> Cond{CONDICIÓN} Cond -- SI --> I[I] I --> Cond Cond -- NO --> Exit(()) </pre>	<pre> mientras CONDICIÓN hacer I1; I2; ...; In; fin_mientras; </pre>
Instrucción Repetir	<pre> graph TD Start(()) --> I[I] I --> Cond{CONDICIÓN} Cond -- SI --> I Cond -- NO --> Exit(()) </pre>	<pre> repetir I1; I2; ...; In; mientras CONDICIÓN; </pre>
Instrucción Para	<pre> graph TD Init[Variable = valor inicial] --> Cond{Variable <= valor_final} Cond -- SI --> Inc[Variable=variable+1] Inc --> I[I] I --> Cond Cond -- NO --> Exit(()) </pre>	<pre> para Vc de Vi a Vf con incremento de In hacer I1; I2; ...; In; finpara; </pre>