



Estructuras de datos

U1- Introducción a las Estructuras de datos

Ever Juárez Guerra

Primavera 2022



Agenda

- Terminología
- Importancia de las estructuras de datos
- Relación entre datos y algoritmos



Terminología

Antecedentes

Las proezas tecnológicas que contribuyen en prácticamente todas las áreas de estudio y el quehacer humano son creadas por personas creativas que desarrollan las habilidades superiores del pensamiento, como:

- el razonamiento abstracto,
- el pensamiento crítico,
- la resolución de problemas y
- la toma de decisiones, entre otras habilidades.

Pensamiento computacional

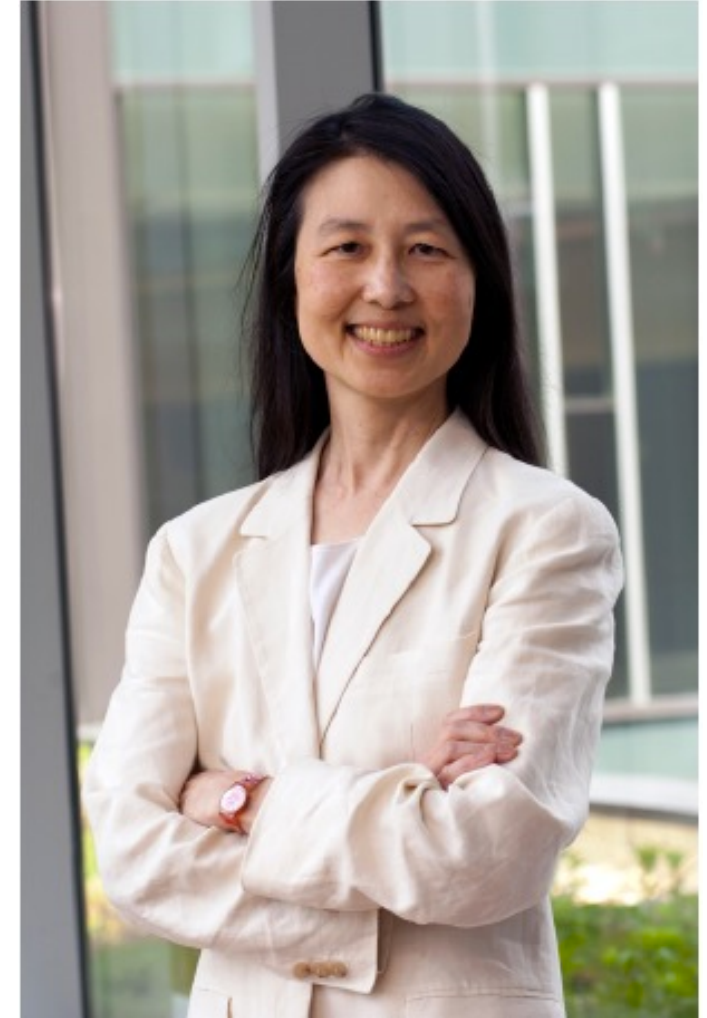
La Fundación Nacional para la Ciencia (NSF), por medio del Sociedad Internacional para la Tecnología en la Educación (ISTE) y la Asociación de profesores de informática (CSTA), impulsa activamente un nuevo enfoque de enseñanza para que en todos los niveles de educación se incluya el **Pensamiento Computacional (Computational Thinking)**.

Este nuevo enfoque busca promover en la educación el desarrollo de habilidades de pensamiento que conduzcan a la formación de personas orientadas a la creatividad y a la innovación.

Pensamiento computacional

La investigadora Jeannette Wing define el **Pensamiento Computacional (PC)** como:

“los procesos de pensamiento involucrados en la formulación de problemas y representación de sus soluciones, de manera que dichas soluciones puedan ser ejecutadas efectivamente por un agente de procesamiento de información (humano, computadora o combinaciones de humanos y computadoras)”.



Componentes del pensamiento computacional

El objetivo del PC es desarrollar sistemáticamente las habilidades del pensamiento de orden superior, como:

- el razonamiento abstracto,
- el pensamiento crítico y
- la resolución de problemas, con base en los conceptos de la computación.

Además, potenciar el aprovechamiento del poder de cálculo que tienen las computadoras actualmente para innovar y volverlo una herramienta científica.

Abstracción

El término abstracción se entiende popularmente en algunas ocasiones con un significado contrario al que tiene en realidad. Cuando algo parece excesivamente complejo o de difícil entendimiento (por ejemplo, una obra de arte), se suele expresar, “es muy abstracto”. Sin embargo, por el contrario, **la abstracción es el filtro utilizado para quedarse con lo que se considera esencial**, eliminando toda la complejidad innecesaria.

La abstracción es la habilidad que le permite al ser humano combatir la complejidad al considerar sólo lo esencial del objeto o fenómeno que se esté analizando.

Generalización

La generalización es el proceso de formular conceptos genéricos a través de la extracción de cualidades comunes de ejemplos concretos.

De manera semejante, el concepto general de “árbol” se establece a partir de las cualidades comunes que lo caracterizan. Por ejemplo, las propiedades comunes de un ahuehuete, un fresno, un ocote y un eucalipto son el tiempo de vida, el número de troncos y la altura a la que se ramifican.

Un tejo de fuego (arbusto) no puede generalizarse al concepto de árbol porque se ramifica desde su base; es decir, no tiene altura y, por esta razón, tiene varios troncos.



(a) Ahuehuete

(b) Fresno



(c) Ocote

(d) Eucalipto

Eliminación de detalles

La eliminación de los detalles es el proceso de dejar fuera de consideración una o más propiedades de un objeto con la finalidad de enfocarse sólo en algunas propiedades. Es decir, únicamente se capturan las propiedades que son relevantes para un determinado problema o área de estudio.

En función de lo que se pretende representar, la abstracción indica qué debe ser considerado relevante, qué detalles deben eliminarse, cuál es el núcleo o esencia y hasta qué punto debe simplificarse una representación.

Información

La información es uno de los recursos más valiosos e importantes en cualquier actividad de nuestra vida diaria o en el quehacer profesional. Sin embargo, es común que este término se confunda con el de datos.

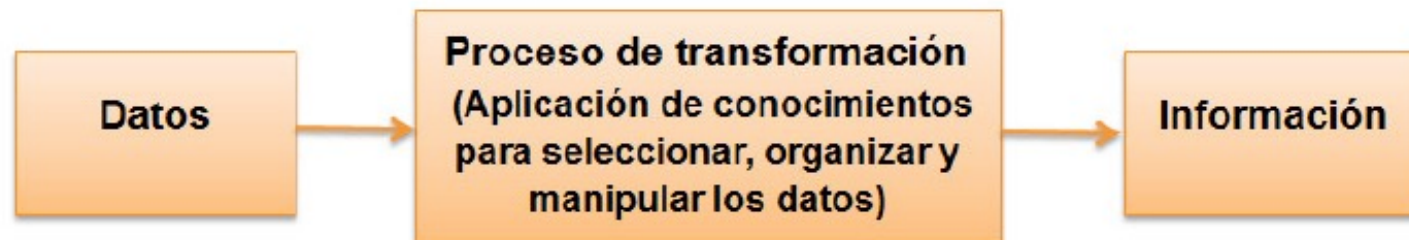
Los datos son la representación de realidades concretas en su estado primario. Por ejemplo, el peso de un vehículo o el nombre de un empleado. Para representar esas realidades es posible usar varios tipos de datos como son: números, letras, imágenes, etc. Los datos no tienen capacidad de comunicar un significado por sí mismos.

Transformación de datos en información

La transformación de datos en información es un proceso o serie de tareas lógicamente relacionadas entre sí y ejecutadas con el fin de producir un resultado definido. El proceso para definir relaciones entre datos requiere de conocimiento.

Por ejemplo, parte del conocimiento necesario para hacer una escalera se fundamenta en la comprensión del hecho de que los peldaños deben colocarse horizontalmente y los postes en forma vertical.

En consecuencia, la información puede considerarse como datos provisto de mayor utilidad mediante la aplicación del conocimiento.



Características de la información

La información debe tener ciertas características para que sea valiosa. Las características que la información debe tener son las siguientes:

- **Exacta:** La información exacta carece de errores. La información inexacta se genera cuando se insertan datos inexactos en el proceso de transformación.
- **Completa:** La información completa contiene todos los datos necesarios. Por ejemplo, un informe de inversión que no incluyera todos los costos estaría incompleto.
- **Simple:** La información debe enfocarse en lo esencial. Un exceso de información hace que la información sea compleja y difícil de identificar lo verdaderamente importante.
- **Confiable:** La información confiable tiene un sólido porcentaje de seguridad respecto a su veracidad. Depende de diversos factores como el método de recolección de datos o la fuente de información. Un rumor de fuente anónima sobre la posibilidad de incremento en los precios del petróleo no sería confiable.

Características de la información

- **Verificable:** La información debe comprobarse si es correcta, quizá mediante la consulta de diversas fuentes al respecto.
- **Oportuna:** La información oportuna es la que se recibe en el momento adecuado o conveniente.
- **Segura:** La información debe estar protegida contra el acceso a usuarios no autorizados.
- **Accesible:** La información debe ser accesible para los usuarios autorizados en el formato adecuado y en el momento correcto.
- **Económica:** La información debe tener un costo relativamente bajo. El costo de producir la información debe ser evaluado frente al valor que provee al usuario.
- **Pertinente:** La información pertinente es adecuada en un contexto determinado. Información acerca de la reducción del precio del limón no sería pertinente para una empresa que fabrica automóviles.

Algoritmos

La palabra algoritmo se deriva de la traducción al latín de la palabra *Alkhô-warîzmi*, nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX.

Un algoritmo es un método para resolver un problema mediante una serie de pasos precisos, definidos y finitos.

Características de un algoritmo

Debe ser/tener:

- Simple
- Datos de entrada.
- Información de salida.
- Proceso definido (claro). Cada paso o decisión debe estar definido con precisión y sin ambigüedad.
- Debe tener un orden lógico (indica el orden de realización de cada paso).
- Preciso y exacto (si se sigue dos veces o más, obtiene el mismo resultado cada vez).
- Tener un principio y un fin (finito). Después de un número finito de pasos, un algoritmo no debería representar un proceso interminable o infinito.
- Efectividad. Debe hacer lo que se supone que debe hacer y no otra cosa.

Diseño de un algoritmo

Las dos herramientas más utilizadas comúnmente para diseñar algoritmos son:

- **diagramas de flujo**, y
- **pseudocódigos**.

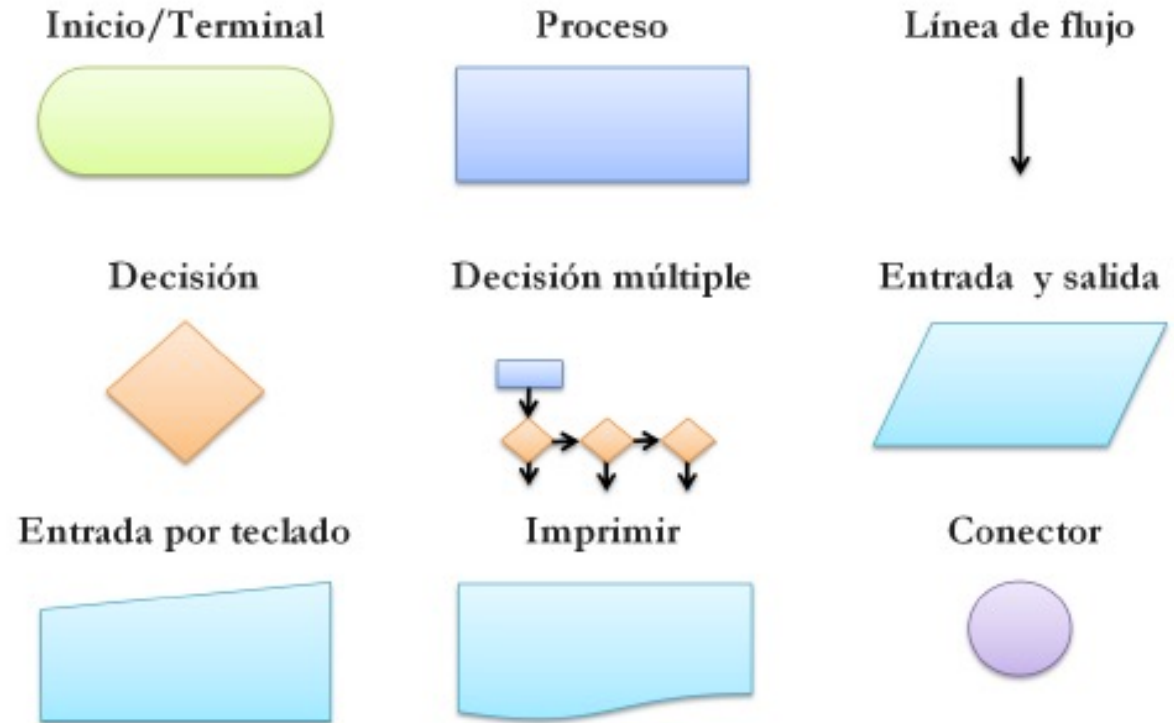
Los **diagramas de flujo** son la representación visual de cada paso del algoritmo por medio de símbolos que representan las operaciones ejecutadas sobre los datos. Hay símbolos aceptados como estándar, a partir de las propuestas de organizaciones como: *American National Standards Institute (ANSI)* y la *International Organization for Standardization (ISO)*.

El **pseudocódigo** es una descripción informal de alto nivel de un algoritmo que utiliza las convenciones de un lenguaje de programación real. El pseudocódigo está diseñado para que el algoritmo sea leído por un humano. Por esta razón, el pseudocódigo se complementa, donde sea conveniente, con descripciones detalladas en lenguaje natural, o con notación matemática.


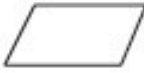

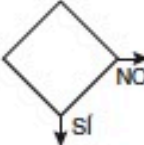
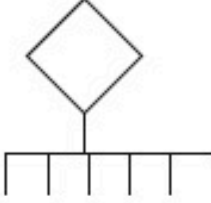

Símbolos básicos de los diagramas de flujo

Los diagramas de flujo se usan para introducir a los estudiantes en el desarrollo de algoritmos por su facilidad de lectura.




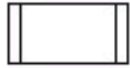



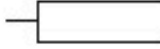
Una desventaja es que abarcan demasiado espacio y su construcción es laboriosa, por esta razón, los diagramas de flujo se usan principalmente para representar algoritmos pequeños.



Símbolos básicos de los 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).

Símbolos básicos de los 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).

Características del pseudocódigo

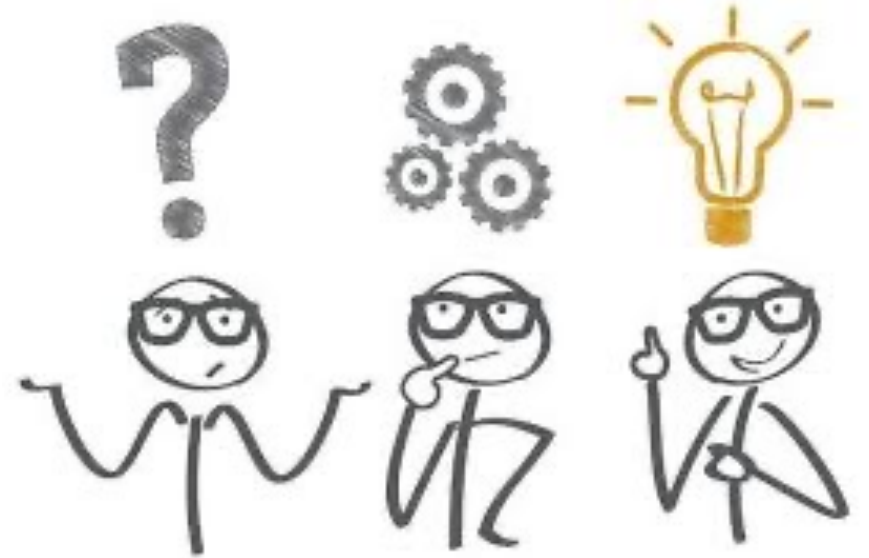
En el pseudocódigo no hay una sintaxis estándar y por lo general, no obedece a las reglas de sintaxis de ningún lenguaje de programación.

El pseudocódigo tiene las siguientes ventajas sobre los diagramas de flujo:

- El espacio utilizado en la descripción del problema es menor.
- Las operaciones complejas se representan de forma sencilla.
- El pseudocódigo es sencillo de trasladar a un lenguaje de programación.
- Las reglas de sangrías permiten observar claramente los niveles en la estructura del algoritmo.

Fases en la solución de problemas

1. Definición del problema.
2. Análisis del problema.
3. Diseño de la solución del problema.
4. Codificación.
5. Compilación, prueba y depuración.
6. Documentación.
7. Implantación del programa.
8. Mantenimiento del programa.





Ejemplos de algoritmos

Algoritmo: Lavarnos los dientes

1. Inicio
2. Tomar la crema dental
3. Destapar la crema dental
4. Tomar el cepillo de dientes
5. Aplicar crema dental al cepillo
6. Tapar la crema dental
7. Abrir la llave del lavamanos
8. Remojar el cepillo con la crema dental
9. Cerrar la llave del lavamanos
10. Frotar los dientes con el cepillo
11. Abrir la llave del lavamanos
12. Enjuagarse la boca
13. Enjuagar el cepillo
14. Cerrar la llave del lavamanos
15. Secarse la cara y las manos con una toalla
16. Fin



¿Qué le hace falta?

Algoritmo: Cambiar la llanta de un vehículo

Instrucciones ordenar los pasos:

- _6_ Aflojar las tuercas de la llanta mientras el auto está en el suelo. Para girarlas, hazlo en sentido contrario a las manecillas de reloj. No quitarlas en este paso.
- _5_ Colocar una palanca. Si se cambia la llanta delantera, colocar la palanca en la parte trasera, y si vas a cambiar la llanta trasera, colocar la palanca en la llanta delantera
- _1_ Inicio
- _8_ Una vez levantado el vehículo, quita las tuercas y retira la llanta dañada.
- _4_ Colocar los triángulos de seguridad a la distancia establecida, usar chaleco reflejante.
- _3_ Buscar la herramienta necesarias (gato hidráulico, otros) y llanta de refacción en el auto
- _11_ Retira el gato hidráulico y herramientas, llanta dañada, guardar.
- _2_ Asegurar el vehículo con el freno de mano y encender las intermitentes.
- _9_ Coloca la llanta de refacción en el centro y procede a poner las tuercas
- _12_ Fin
- _10_ Baja el vehículo y termina de apretar las tuercas o tornillos con la llave de cruz
- _7_ Coloca el gato hidráulico para levantar el vehículo.

Algoritmo: Acciones cuando vas a la escuela

4 Te tomas 5 minutos más
9 Te lavas el cabello
7 Te levantas
11 Te secas
14 Te lavas los dientes
3 Apagas el despertador
19 Fin
18 Dices "En la escuela la hago"
13 Te vistes.

16 Sales de tu casa corriendo
10 Te lavas el cuerpo
15 Te enjuagas la boca
6 Te despiertas como rayo
17 En el camino recuerdas que hay tarea
12 Buscas tu ropa
5 Te das cuenta que no fueron 5 minutos, si no 20.
8 Te metes a bañar rápido.
2 Te despiertas
1 Inicio

Problema a solucionar

La escritura de algoritmos para realizar operaciones sencillas de conteo es una de las primeras cosas que una computadora puede aprender.

Supongamos que se proporciona una secuencia de números, tales como:

5 3 0 2 4 4 0 0 2 3 6 0 2

y desea contar e imprimir el número de ceros de la secuencia.

Realizar el algoritmo en diagrama de flujo y en pseudocódigo.

Solución: Conteo de ceros

El algoritmo es muy sencillo, ya que sólo basta leer los números de izquierda a derecha, mientras se cuentan los ceros. Utiliza como variable la palabra NUMERO para los números que se examinan y TOTAL para el número de ceros encontrados.

Los pasos a seguir son:

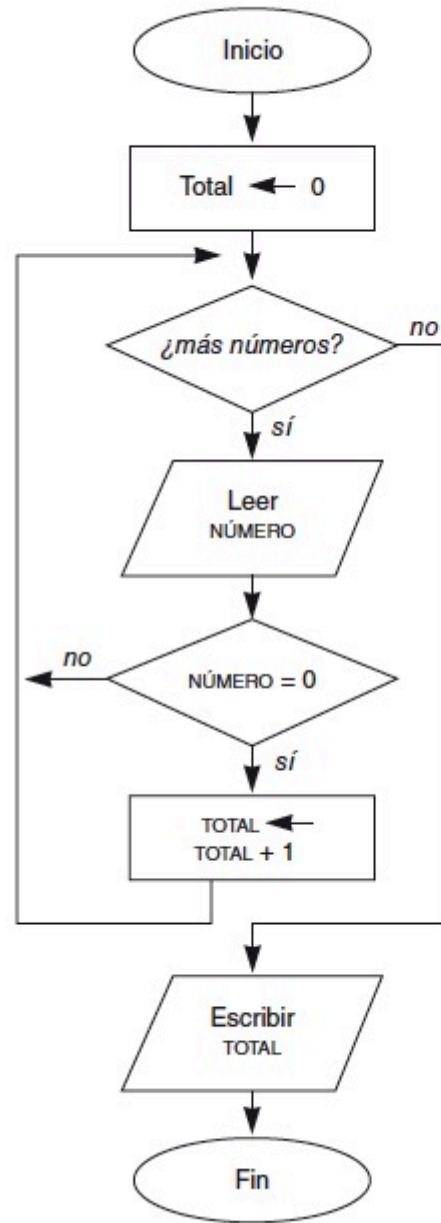
5 3 0 2 4 4 0 0 2 3 6 0 2

1. Inicio
2. Establecer TOTAL a cero.
3. ¿Hay más números a examinar?
4. Si no quedan números, imprimir el valor de TOTAL y fin.
5. Si existen más números, ejecutar los pasos 6 a 9.
6. Leer el siguiente número y asigna su valor a la variable NUMERO.
7. Si $\text{NUMERO} = 0$, incrementar TOTAL en 1.
8. Si $\text{NUMERO} \neq 0$, no modificar TOTAL.
9. Regresar al paso 3.
10. Fin

Solución. Conteo de ceros

5 3 0 2 4 4 0 0 2 3 6 0 2

Diagrama de flujo



Pseudocódigo

entero: numero, total

caracter: mas Datos;

inicio

escribir ('Cuenta de ceros leídos del teclado')

mas Datos ← 'S';

total ← 0

mientras (mas Datos = 'S') o (mas Datos = 's') **hacer**

 leer (numero)

si (numero = 0)

 total ← total + 1

fin si

 escribir ('Mas números 'S/N'')

 leer (mas Datos)

fin mientras

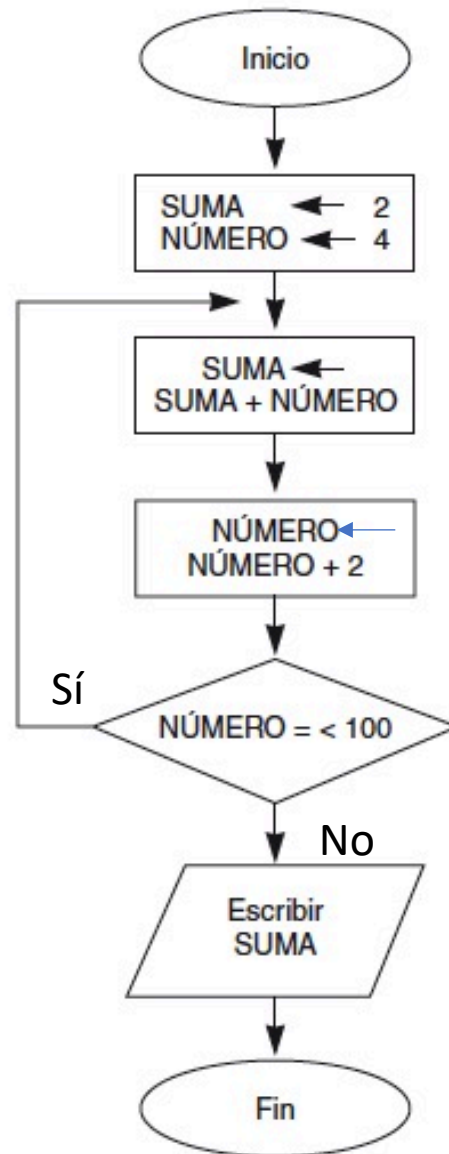
escribir ('total de ceros =', total)

fin

Ejemplo: Suma
de los números
pares
comprendidos
entre 2 y 100.

2, 4, 6, 8, 10, 12..., 100

Diagrama de flujo



Pseudocódigo

```
entero:  numero, Suma
Suma ← 2
numero ← 4
mientras (numero ≤ 100) hacer
    suma ← suma + numero
    numero ← numero + 2
fin mientras
escribe ('Suma pares entre 2 y 100 =', suma)
```

Ejemplo:

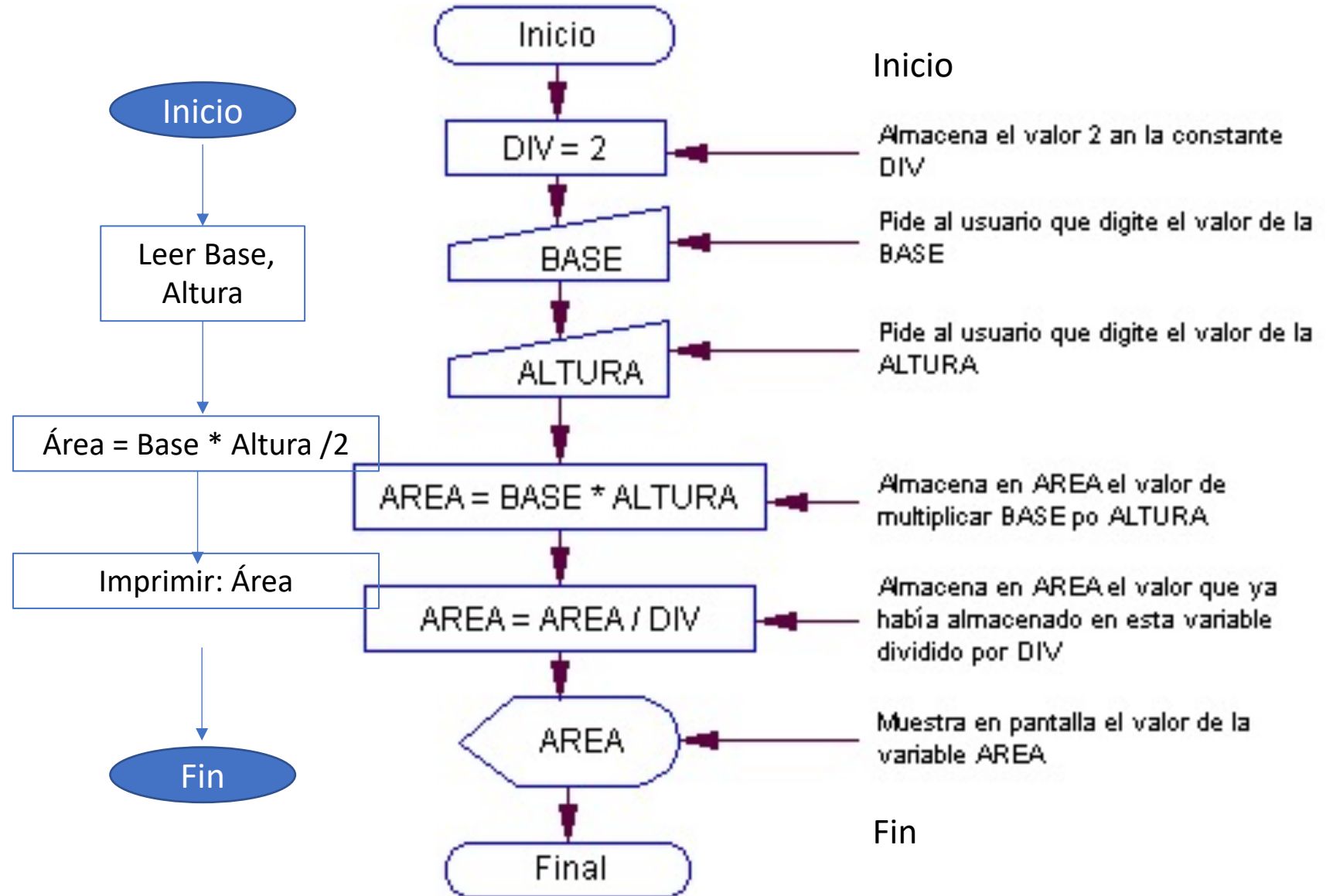
Elaborar un algoritmo en pseudocódigo y diagrama de flujo para calcular el área de cualquier triángulo rectángulo y presentar el resultado en pantalla.

Ejemplo: Solución área del Triángulo rectángulo

PSEUDOCÓDIGO

Inicio
Leer: Base, Altura
Calcular área: $\text{Área} = \text{Base} * \text{Altura} / 2$
Imprimir: Área
Fin

DIAGRAMA DE FLUJO



LISTA DE PASOS

Inicio

Almacena el valor 2 en la constante DIV

Pide al usuario que digite el valor de la BASE

Pide al usuario que digite el valor de la ALTURA

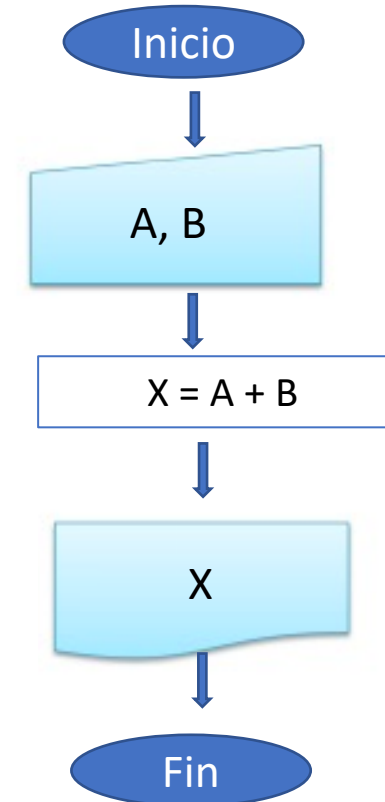
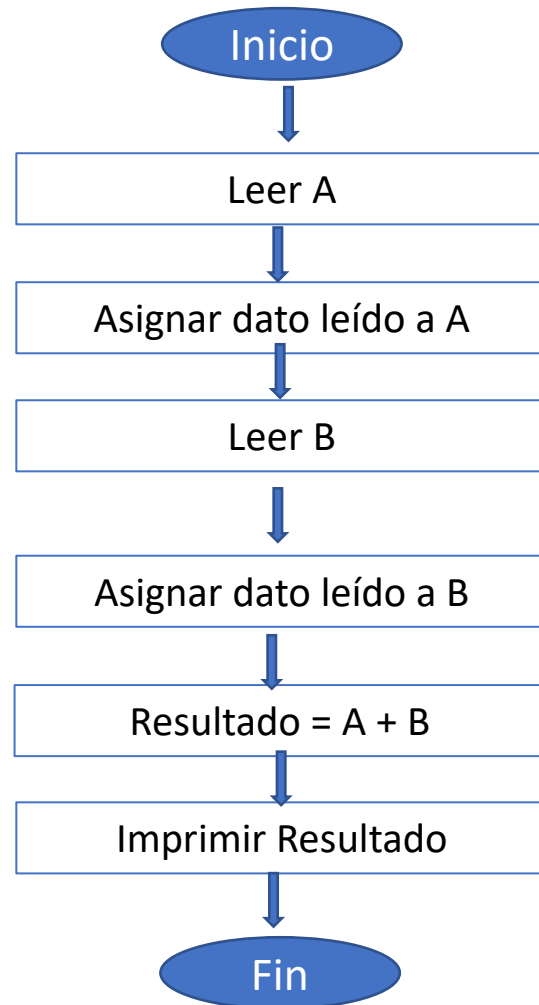
Almacena en AREA el valor de multiplicar BASE por ALTURA

Almacena en AREA el valor que ya había almacenado en esta variable dividido por DIV

Muestra en pantalla el valor de la variable AREA

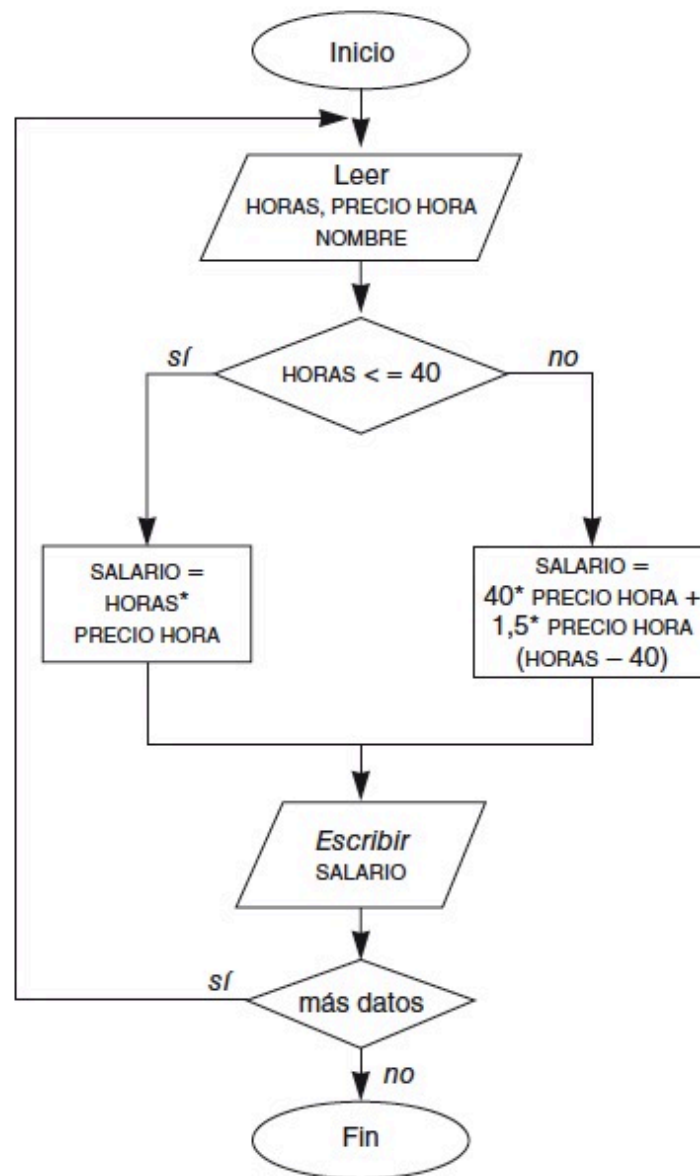
Fin

Diagramas de flujo de un algoritmo que suma dos números



Problema:
Analiza el
siguiente
algoritmo y
explica el
problema que
soluciona

Diagrama de flujo



Pseudocódigo

```
real: horas, precioHora, salario  
cadena: nombre  
caracter: masDatos
```

inicio

```
escribir('Introducir horas, precio hora  
y nombre')
```

repetir

```
    escribir ('Nombre')  
    leer (Nombre)  
    escribir ('Horas trabajadas')  
    leer (horas)  
    escribir ('Precio hora')  
    leer (precio Hora)
```

si (horas <= 40) **entonces**

```
    Salario ← horas * precioHora
```

sino

```
    Salario ← 40 * precioHora +  
              1.5 * (horas - 40) *  
              preciohora
```

fin si

```
    escribir ('Salario de', nombre, salario)
```

```
    escribir ('Mas trabajadores S/N')
```

```
    leer (masDatos)
```

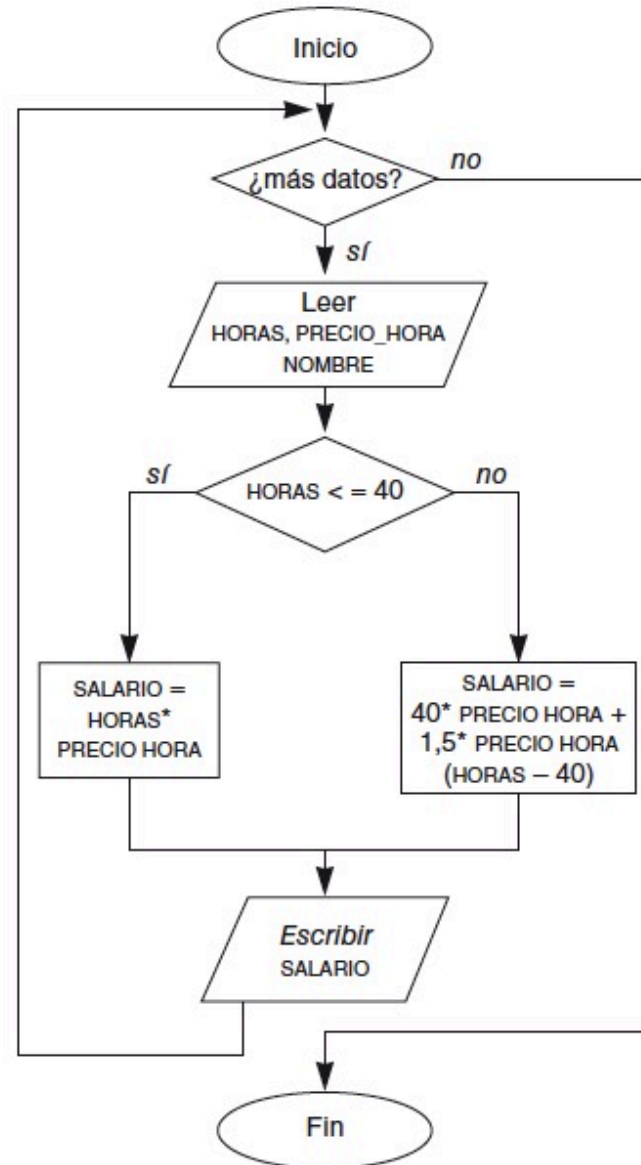
hasta masDatos = 'N'

fin

Problema:

¿cuál es la
diferencia con el
algoritmo
anterior?
Explica

Diagrama de flujo



Pseudocódigo

```
real: horas, precioHora, salario
cadena: nombre
caracter: masDatos

inicio
masDatos ← 'S';
escribir ('Introducir horas, precio hora y nombre')

mientras (masDatos = 'S' o masDatos = 's')
hacer
    escribir ('Nombre')
    leer (nombre)
    escribir ('Horas trabajadas')
    leer (horas)
    escribir ('Precio hora')
    leer (precioHora)
    si horas <= 40 entonces
        salario ← horas * precioHora
    sino
        salario ← 40 * precioHora + 1.5 * (horas - 40) * precioHora
    fin si
    escribir ('Salario de', nombre, salario)
    escribir ('Mas trabajadores (S/N)')
    leer (masDatos)
fin mientras
fin
```



Importancia de las estructuras de datos

Importancia de las estructuras de datos

La programación de computadores a través de lenguajes, inició en la década de los cincuenta con un rudimentario lenguaje llamado lenguaje de maquina (código binario), luego se pasó a ensamblador (nemotécnicos), ambos muy difíciles de entender y escribir.

Con el tiempo estos evolucionaron en una diversidad de lenguajes de alto nivel, fáciles de entender y de escribir por el uso de instrucciones con palabras de nuestro propio lenguaje, eran especializados de acuerdo al campo de aplicación o con propósito general capaces de brindar mejores herramientas para desarrollo de todo tipo de programas.

Importancia de las estructuras de datos

Gran parte de la mejoría en los programas se ha logrado por el establecimiento de técnicas de programación soportadas por los lenguajes, cuyo propósito no es más que el de optimizar costo en la producción de programas, el costo se refiere a la escritura, su mantenimiento y el uso de recursos por parte del programa.

Las técnicas siempre han buscado reducir las instrucciones creando componentes reutilizables, y organizar de mejor forma los datos de tal manera que se puedan implementar algoritmos más eficientes, que disminuyan el tiempo y faciliten el procesamiento.

Importancia de las estructuras de datos

Hasta el inicio de la década de los setentas la programación era muy simple, no había ninguna preocupación por desarrollar programas de alta calidad. En esta misma década son desarrolladas por iniciativa de un grupo de ingenieros un conjunto de técnicas enmarcadas bajo el nombre de **ingeniería de software** que actualmente se considera como una verdadera disciplina de la ingeniería, cuyo fin es proporcionar:

- Mucho más calidad a todas las etapas de desarrollo de un programa
- Resolver un problema correcto
- Entregar una solución a tiempo
- Solución dentro del presupuesto y,
- Sobre todo con una solución de alta calidad

Todo el proceso anterior se convierte en proyecto de sistemas.

Importancia de las estructuras de datos

Las estructuras de datos no son ajenas a todo este proceso evolutivo de la programación por el contrario, son la base para el desarrollo de programas complejos que se diseñan cuidadosamente.

Inicialmente los programas no eran estructurados porque carecían de organización, estructuras de control y datos complejos; estos fueron introducidos a medida que evolucionaban los lenguajes, lo que permitió crear datos compuestos a partir de la agrupación de datos simples por parte de los programadores.

Muchos algoritmos requieren una presentación apropiada de los datos para lograr ser eficientes, esta presentación junto con las operaciones permitidas se llama **estructura de datos** (Mark Allen Weiss: 2006, 37).

Importancia de las estructuras de datos

En la actualidad, la eficiencia de un programa informático va de la mano con las técnicas de programación que se emplean en su desarrollo, partiendo desde la elaboración de diagramas de flujo de datos, hasta la escritura de los códigos para el desarrollo del software.

Lo anterior busca el acceso a los datos de la información de una manera ordenada mediante instrucciones válidas, empleando una secuencia lógica.

La **estructura de datos** se refiere a un conjunto de técnicas que aumentan considerablemente la productividad del programa, reduciendo en elevado grado, el tiempo requerido para escribir, verificar, depurar y mantener los programas.

Tipos de estructuras de datos

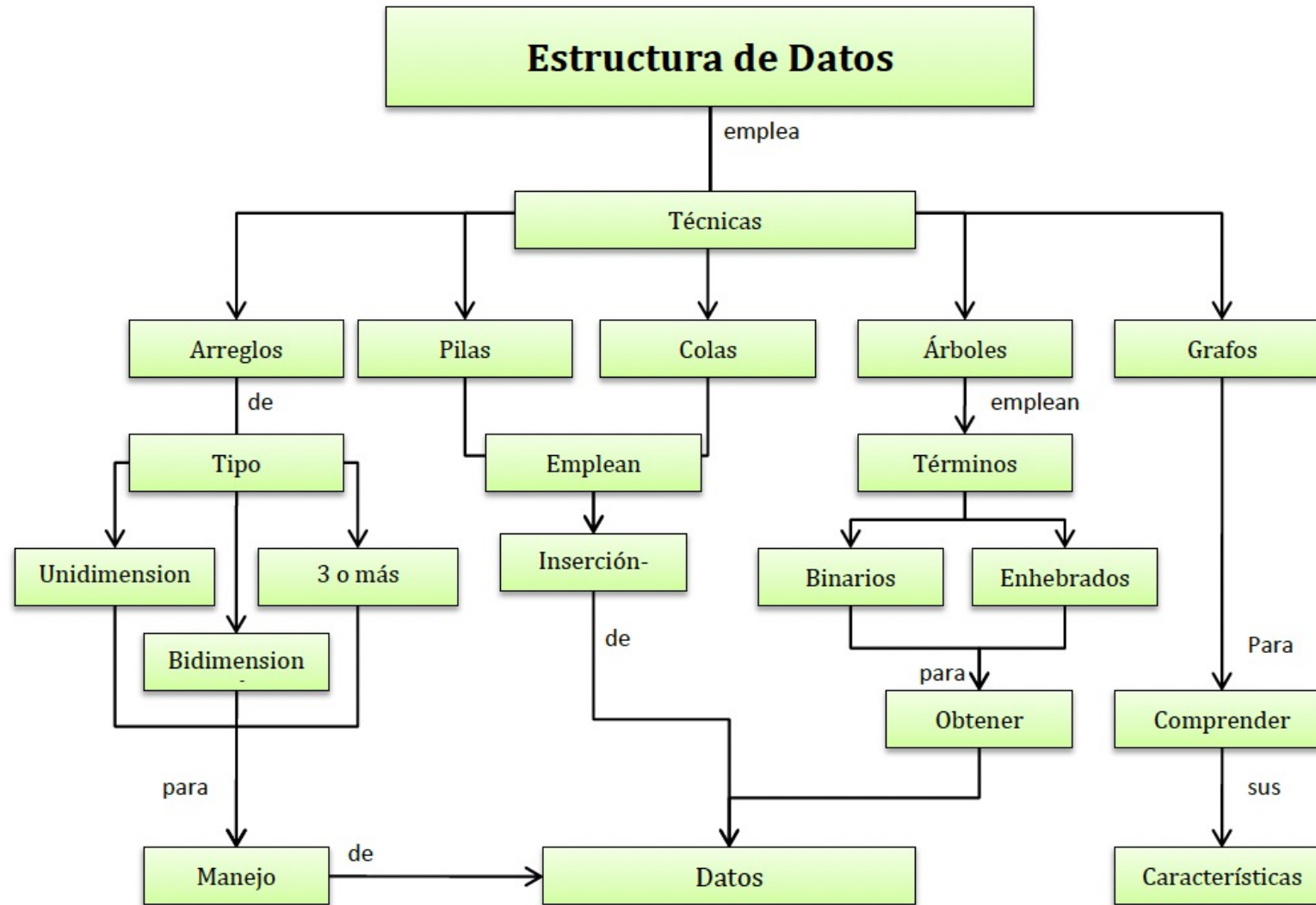
Las estructuras de datos pueden ser de dos tipos:

- Estructuras de datos estáticas (las que tienen un tamaño definido).
- Estructuras de datos dinámicas (en las cuales su tamaño puede ser cambiado en tiempo de ejecución).

De acuerdo a la solución requerida las estructuras de datos se clasifican en:

- Arreglos
- Listas enlazadas
- Pilas
- Colas
- Árboles
- Grafos

Tipos de estructuras de datos



¿Porqué son útiles las estructuras de datos?

Nos permiten lograr un importante objetivo de las técnicas de programación: la reutilización de componentes.

La eficiencia en los sistemas de información está basada en la velocidad de procesamiento, la confiabilidad, la seguridad y el aprovechamiento de los recursos.

La disposición de los datos y el manejo adecuado del espacio de memoria es necesaria para el mejoramiento en los procesos de información, por lo que es muy importante diseñar y utilizar estructuras de datos.

A través de estas se representan y abstraen situaciones reales de la vida, usándolas como herramientas para la evaluación de soluciones óptimas de problemas y necesidades en el complejo mundo de la informática.



Relación entre datos y algoritmos

Relación entre datos y algoritmos

Algoritmo

Un algoritmo, en su concepción general, tiene una *entrada*, un *proceso* y una *salida*. La entrada vendría siendo un **dato**, un valor inicial para comenzar el proceso.

El proceso es el núcleo del **algoritmo**, normalmente utiliza el dato de entrada para modificarlo o para decidir que hacer. La salida entonces es el resultado final del proceso, lo que él genera.

Es importante notar que tanto la entrada como la salida son opcionales, esto quiere decir que el algoritmo puede ejecutarse sin un valor inicial, así mismo y puede no generar un resultado al terminar.

Relación entre datos y algoritmos

Datos

Los datos son los *valores* a utilizar dentro del **algoritmo**, datos pueden ser valores de entrada del algoritmo, o el resultado del mismo; el algoritmo puede usar los datos para tomar decisiones, para modificarlos, para almacenarlos en una base de datos, puede incluso leer datos desde una base de datos, mostrarlos en pantalla (si está disponible), enviarlos a otro algoritmo/proceso/computador.

Relación entre datos y algoritmos

Variable

Una variable es una estructura de datos (simple o compleja) en la que se puede *almacenar* un dato o múltiples datos. Los algoritmos utilizan las variables para poder *acceder y manipular* los datos.

Por ejemplo, si consideramos la suma de dos números:

$$5 + 3 = 8$$

5, 3 y 8 son ***datos***, donde: 5 y 3 los datos de entrada, 8 el dato de salida.

La suma (+) vendría siendo el ***algoritmo***.

Y en este caso particular *no hay variables*. Entonces, ¿Para que las necesitamos?

Relación entre datos y algoritmos

Variables, ¿para qué las necesitamos?

El algoritmo anterior, solo serviría para sumar 5 y 3, exclusivamente. Por lo tanto, si queremos que el mismo algoritmo sirva para otras sumas tendremos que incorporar variables de la siguiente manera:

$$A + B = C$$

Ahora, A y B, pueden tomar cualquier valor numérico y entonces C obtendría el resultado de la suma de los valores. En computación sería:

$$C = A + B$$

Entonces, A y B son variables que contienen los datos de entrada; el algoritmo realiza la suma, generando un nuevo dato, y la almacena en C.

¿Preguntas?



Referencias

Ayala San Martín, G. Algoritmos y programación. Fundación Universidad de las Américas Puebla, 2018.

Zapotecatl López J.L., Introducción al pensamiento computacional: conceptos básicos para todos. Academia Mexicana de Computación, A.C. Conacyt , 2018.

Joyanes Aguilar, Luis. Fundamentos de programación. Algoritmos, estructura de datos y objetos. Cuarta edición, Mc. Graw Hill, 2008.

López García, C. Algoritmos y Programación. Fundación Gabriel Piedrahita Uribe. Segunda Edición. 2009.

Wing, J. M., Computational thinking and thinking about computing. Conference on Institute for Human & Machine Cognition, 2009.