



UNIVERSIDAD TECNOLÓGICA DE PANAMÁ
FACULTAD DE INGENIERÍA DE SISTEMAS
PROGRAMACIÓN APLICADA

Contenido

Conceptos de Algoritmos	1
Algoritmo.....	1
Características de un Algoritmo	1
Secciones de un Algoritmo	2
Pseudocódigo	2
Diagrama de Flujo	2
Conceptos Fundamentales.....	7
Tipos de Datos.....	7
Asignación No. 1.....	10
Expresión.....	11
Multiplicación.....	11
Promedio.....	11
Importe de la compra.....	11
Cuadrado y cubo	11

Conceptos de Algoritmos

Algoritmo

En nuestro día a día nos enfrentamos a diversas situaciones, algunas de las cuales requieren que se implementen algunas acciones para ser resueltas.

Diseñar y desarrollar software conlleva realizar una serie de pasos desde que identificamos el problema hasta que se implementa la solución “software”.

Cuando estamos analizando un problema para buscar la solución elaboramos una serie de pasos en un orden específico, esta serie de pasos lo llamamos algoritmo. Osvaldo Cairo define algoritmo así:

“Formalmente definimos un algoritmo como un conjunto de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema”.

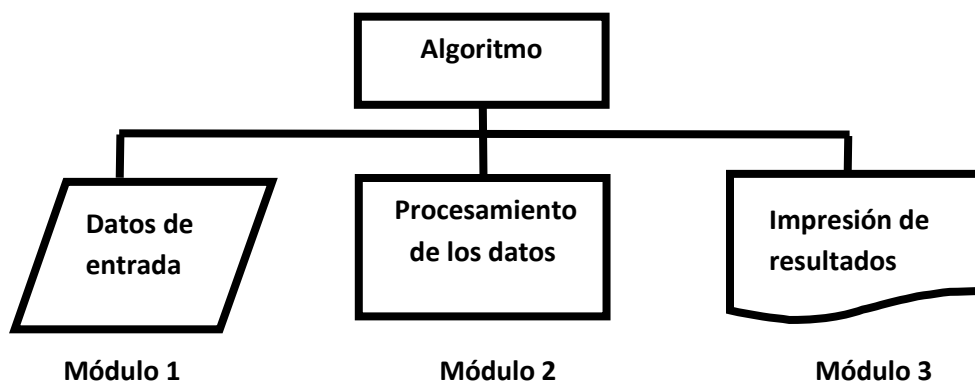
Diariamente aplicamos en nuestras vidas los algoritmos de forma inadvertida en casos cotidianos, abrir una puerta, subir al auto, etc.

Características de un Algoritmo

Los algoritmos tienen las siguientes características.

1. **Precisión:** los pasos a seguir en el algoritmo deben ser precisados claramente.
2. **Determinismo:** el algoritmo, dado un conjunto de datos idénticos de entrada, siempre debe arrojar los mismos resultados.
3. **Finitud:** el algoritmo, independientemente de la complejidad del mismo, siempre debe ser de longitud finita.

Secciones de un Algoritmo



Pseudocódigo

El pseudocódigo (o falso lenguaje) es comúnmente utilizado por los programadores para omitir secciones de código o para dar una explicación del paradigma que tomó el mismo programador para hacer sus códigos, esto quiere decir que el pseudocódigo no es programable sino facilita la programación.

El principal objetivo del pseudocódigo es el de representar la solución a un algoritmo de la forma más detallada posible.

Cabe señalar que el pseudocódigo es independiente al Lenguaje de Programación que se utilice para implementar la solución.



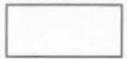
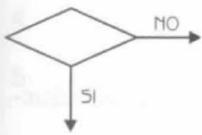
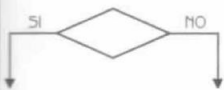






Reglas para la construcción de un pseudocódigo:

1. Debe tener un inicio
2. Las líneas deben ir numeradas.
3. Utilizar verbos como Escribir, Leer, Hacer
4. Debe tener un fin

Diagrama de Flujo

Representa la esquematización gráfica de un algoritmo. Muestra los pasos o procesos a seguir para la solución de un problema.

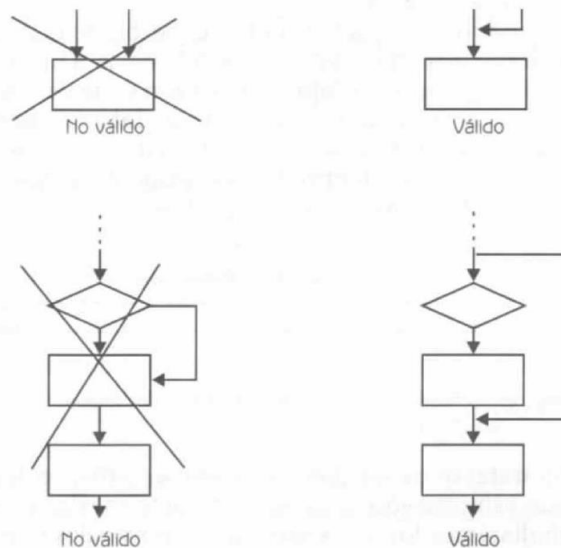
Símbolos utilizados en los Diagramas de Flujo

Representación del Símbolo	Explicación del Símbolo
	Símbolo utilizado para marcar el <i>inicio</i> y el <i>fin</i> del diagrama de flujo.
	Símbolo utilizado para introducir los datos de entrada. Expresa <i>lectura</i> .
	Símbolo utilizado para representar un <i>proceso</i> . En su interior se expresan asignaciones, operaciones aritméticas, cambios de valor de celdas en memoria, etc.
	Símbolo utilizado para representar una decisión. En su interior se almacena una condición, y dependiendo del resultado de la evaluación de la misma se sigue por una de las ramas o caminos alternativos. Este símbolo se utiliza en la estructura selectiva <i>si entonces</i> que estudiaremos en el siguiente capítulo, y en las estructuras repetitivas <i>repetir</i> y <i>mientras</i> que analizaremos en el capítulo 3.
	Símbolo utilizado para representar la estructura selectiva doble <i>si entonces/sino</i> . En su interior se almacena una condición. Si el resultado es verdadero se continúa por el camino de la izquierda, y si es falso por el camino de la derecha.
	Símbolo utilizado para representar una decisión múltiple. En su interior se almacena un selector, y dependiendo del valor de dicho selector se sigue por una de las ramas o caminos alternativos. Este símbolo se utiliza en la estructura selectiva <i>si múltiple</i> , que analizaremos en el siguiente capítulo.
	Símbolo utilizado para representar la impresión de un resultado. Expresa <i>escritura</i> .
	Símbolos utilizados para expresar la dirección del flujo del diagrama.
	Símbolo utilizado para expresar conexión dentro de una misma página.
	Símbolo utilizado para expresar conexión entre páginas diferentes.
	Símbolo utilizado para expresar un módulo de un problema. En realidad expresa que para continuar con el flujo normal del diagrama debemos primero resolver el subproblema que enuncia en su interior.

Reglas para la construcción de un diagrama de flujo:

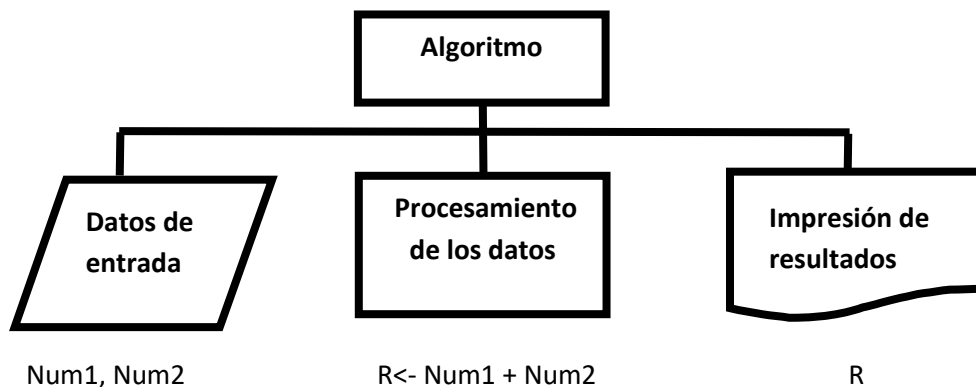
1. Todo diagrama de flujo debe tener un inicio y un fin.
2. Las líneas utilizadas para indicar la dirección del flujo del diagrama deben ser rectas, verticales y horizontales.
3. Todas las líneas utilizadas para indicar la dirección del flujo del diagrama deben estar conectadas.

4. El diagrama de flujo debe ser construido de arriba hacia abajo y de izquierda a derecha.
5. La notación utilizada en el diagrama de flujo debe ser independiente del lenguaje de programación. La solución presentada en el diagrama puede escribirse posteriormente y fácilmente en diferentes lenguajes de programación.
6. Es conveniente cuando realizamos una tarea compleja poner comentarios que expresen o ayuden a entender lo que hicimos.
7. Si el diagrama de flujo requiera más de una hoja para su construcción, debemos utilizar los conectores adecuados y enumerar las páginas convenientemente.
8. No puede llegar más de una línea a un símbolo.



Ejemplo:

Se desea un programa que lea dos números enteros y escriba el resultado de la suma.



Diseño de Pantalla

Programa para la Suma de Dos Números Enteros

Introducir el primer número entero: _____

Introducir el segundo número entero: _____

El resultado de la suma es: _____

Diagrama de Flujo	Pseudocódigo	Sintaxis en C
<pre> graph TD Inicio([Inicio]) --> Decl[Num1, Num2, R] Decl --> Title[/Programa para sumar dos números enteros/] Title --> Input1[Introducir el primer número entero] Input1 --> Read1[/Num1/] Read1 --> Input2[Introducir el segundo número entero] Input2 --> Read2[/Num2/] Read2 --> Process[R<- Num1 + Num2] Process --> Output[/El resultado de la suma es, R/] Output --> Fin([Fin]) </pre>	<ol style="list-style-type: none"> 1. Inicio 2. {Se declaran las variables Num1 tipo de dato simple, número entero Num2 tipo de dato simple, número entero R tipo de dato simple, número entero} 3. Escribir "Programa para sumar dos números enteros" 4. Escribir "Introducir el primer número entero" 5. Leer Num1 6. Escribir "Introducir el segundo número entero" 7. Leer Num2 8. Hacer R <- Num1 + Num2 9. Escribir "El resultado de la suma es", R 10. Fin 	<pre> #include<stdio.h> /* Programa para sumar*/ void main() { /* Se declaran las variables*/ int Num1, Num2, R=0; printf("Programa para sumar dos números enteros \n "); printf("Introducir el primer número entero \n "); scanf("%d",&Num1); printf("Introducir el segundo número entero \n "); scanf("%d",&Num2); R= Num1 + Num2; printf("El resultado de la suma es:%d \n ", R); } </pre>

Prueba de Escritorio:

Entrada		Proceso	Salida
Num1	Num2	$R \leftarrow \text{Num1} + \text{Num2}$	El resultado del suma es R
1	2	$R \leftarrow 1 + 2$ $R \leftarrow 3$	El resultado del suma es 3
4	8	$R \leftarrow 4 + 8$ $R \leftarrow 12$	El resultado del suma es 12

Conceptos Fundamentales

Tipos de Datos

Los datos a procesar por una computadora pueden clasificarse en:

- Simples: los datos simples ocupan sólo una casilla de memoria. Una variable simple hace referencia a un único valor a la vez.
- Estructurados: los datos estructurados se caracterizan por el hecho de que con un nombre (identificador de variable estructurada) se hace referencia a un grupo de casillas de memoria.

Datos numéricos

Dentro de los tipos de datos numéricos encontramos los enteros y los reales.

Datos alfanuméricos

Dentro de los tipos de datos encontramos los tipos de datos carácter y cadena de caracteres.

Datos lógicos

Dentro de este tipo de datos encontramos los booleanos. Son datos que sólo pueden tomar uno de los siguientes valores verdadero o falso, uno (1) o cero (0).

Identificadores

Los datos a procesar por una computadora, ya sean simples o estructurados, deben almacenarse en casillas o celdas de memoria para su posterior utilización. Estas casillas o celdas de memoria (constante o variables) tienen un nombre que permite su identificación.

Llamaremos identificador al nombre que se le da a las casillas de memoria.

Un identificador se forma de acuerdo a ciertas reglas:

- El primer carácter que forma un identificador debe ser una letra.
- Los demás caracteres pueden ser letras, dígitos o el siguiente símbolo especial _
- La longitud del identificador es igual a 7

Constantes

Las constantes son datos que no cambian durante la ejecución de un programa.

Variables

Las variables son datos que pueden cambiar su valor de ejecución durante la ejecución de un programa.

Operadores Aritméticos

Operador Aritmético	Operación	Ejemplo	Resultado
**	Potencia	4**3	64
*	Multiplicación	8.25*7	57.75
/	División	15/4	3.75
+	Suma	125.78 + 62.50	188.28
-	Resta	65.30 - 32.33	32.97
mod	Módulo (residuo)	15 mod 2	1
div	División entera	17 div 3	5

Jerarquía de los operadores aritméticos

Operador	Jerarquía	Operación
**	(mayor)	Potencia
*,/,mod,div	↓	Multiplicación, división, módulo, división entera
+,−	(menor)	Suma, resta

Operadores Relacionales

Operador	Operación	Ejemplo	Resultado
=	Igual que	'hola' = 'lola'	FALSO
< >	Diferente a	'a' < > 'b'	VERDADERO
<	Menor que	7 < 15	VERDADERO
>	Mayor que	22 > 11	VERDADERO
<=	Menor o igual que	15 <= 22	VERDADERO
>=	Mayor o igual que	35 >= 20	VERDADERO

Operadores Lógicos

Operador lógico	Jerarquía	Expresión lógica	Significado
NO	(mayor)	No P	NO P No es cierto que P Es FALSO que P
Y	↓	P y Q	$P \wedge Q$ P sin embargo Q
O	(menor)	P o Q	$P \vee Q$ o P o Q o ambas Mínimo P o Q

Asignación No. 1

Todo trabajo entregado después de la fecha asignada tiene un descuento de 25 puntos por semana de atraso.

Para los siguientes supuestos realizar el análisis.

Entregar:

1. Secciones de un algoritmo 3puntos por caso
2. Diseño de la pantalla 3puntos por caso
3. Diagrama de flujo 3puntos por caso
4. Pseudocódigo y 3puntos por caso
5. Prueba de escritorio 3puntos por caso
6. Codificación en C 5 puntos por caso. Para hacer este punto tiene que tener los 5 puntos anteriores resueltos.

Expresión

1. Se desea un programa que lea dos números enteros y escriba el resultado de la siguiente operación $(A+B)^{**2}/3$.

Nota: Al momento de codificar utilizar la librería math.h y la función pow

Multiplicación

2. Se desea un programa que lea dos números enteros y escriba el resultado de la multiplicación.

Promedio

3. Se desea un programa que lea dos notas parciales y escriba el promedio de las notas parciales.

Importe de la compra

4. Se desea un programa que lea la cantidad comprada de un producto y su precio. El programa debe escribir el importe a pagar $(CANT*PRECIO) * 1.07$.

Cuadrado y cubo

5. Se dese un programa que lea un número entero. El programa debe dar como resultado el cuadrado y el cubo del número leído.

Nota: Al momento de codificar utilizar la librería math.h y la función pow