UTN-FRA
INFORMÁTICA I
ING. ELECTRÓNICA

ALGORITMOS DE PROGRAMACIÓN

Parte 1

DEFINICIONES IMPORTANTES

PROGRAMA

- Es el conjunto de instrucciones escritas en algún lenguaje de programación y que ejecutadas secuencialmente, resuelven un problema específico.

LENGUAJE

- Es una serie de símbolos que sirven para transmitir uno o más mensajes (ideas) entre dos entidades diferentes.

PSEUDOCÓDIGO

- Es una mezcla de un lenguaje de programación y el idioma de un país, se utiliza en la programación estructurada para realizar el diseño de un algoritmo.

DIAGRAMA DE FLUJO

- Es la representación gráfica de un algoritmo. Para resolverlo se utilizan distintos tipos de figuras juntos con flechas conectoras que permiten establecer la secuencia de un proceso.

ALGORITMO

- Es una serie de pasos organizados que describe el proceso que se debe seguir, para dar solución a un problema específico.

ALGORITMO

- Los algoritmos se pueden expresar en forma de **fórmulas matemáticas**, por medio de un **diagrama de flujo**, con el la utilización del **pseudocódigo** y con un **lenguaje programación**.

ALGORITMO

- Es un método que para resolverlos, es mediante una serie de pasos precisos, definidos y finitos.
  - **Preciso:** No se presta a interpretaciones ambiguas.
  - **Definido:** Si se siguen 2 o más veces los pasos, se obtiene el mismo resultado.
  - **Finito:** Tiene comienzo y fin; tiene un número determinado de pasos.

Por ejemplo:

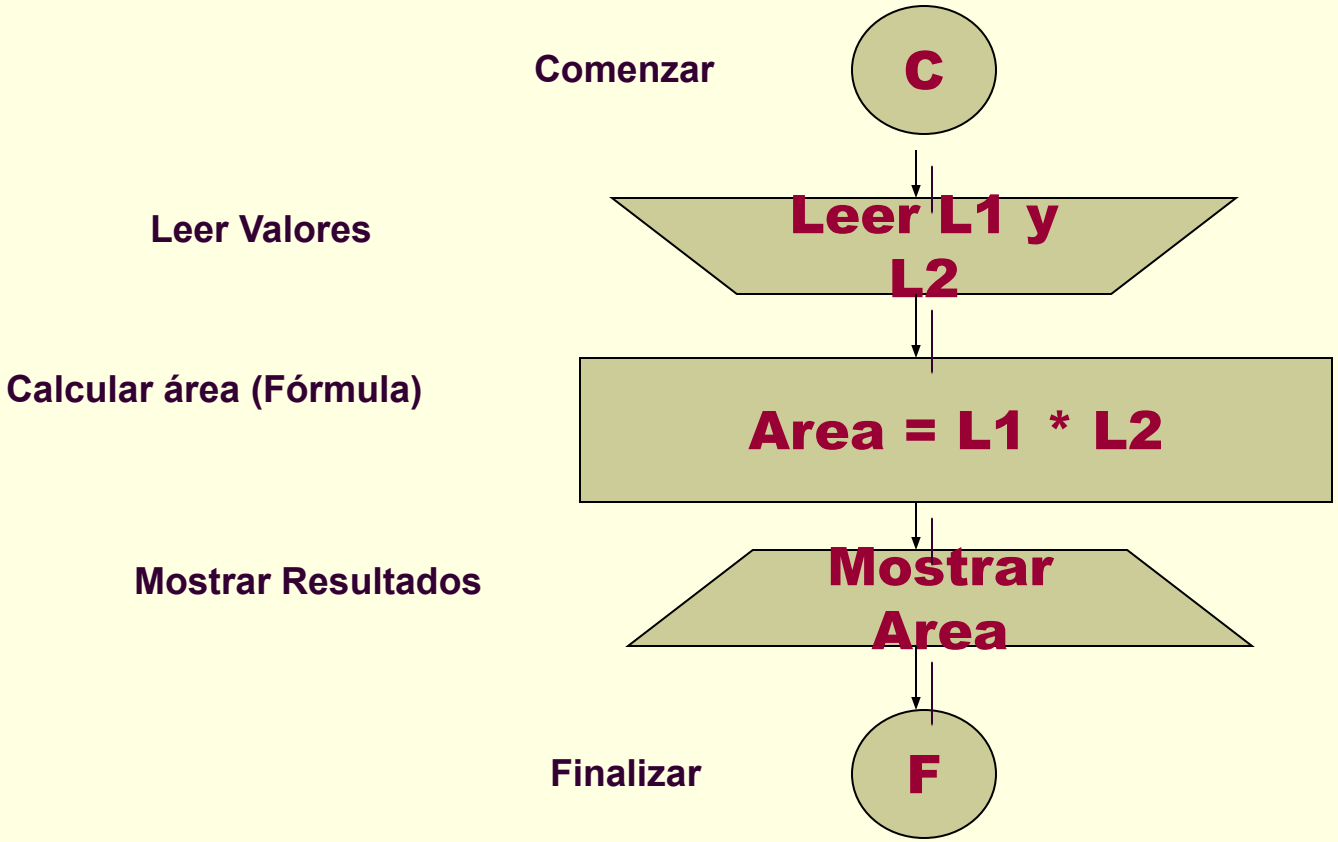
- **Se debe calcular la superficie de un rectángulo, debiéndose ingresar los valores de los lados y calcular el resultado.**
- Para resolución de este problema usaremos las siguientes variables:
  - Para los lados del rectángulo: **L1** y **L2**
  - Para almacenar el resultado: **Area**

Si el Algoritmo lo resolvemos con una **Fórmula Matemática**:

- Le asignamos los valores a los lados del rectángulo:
  - **L1 = 5** y **L2 = 18**
- Calculamos la superficie:
  - **Area = L1 * L2**

"El resultado del proceso es igual a **90**"

Si el Algoritmo lo resolvemos con un **Diagrama de Flujo**:



En cambio si resolvemos el Algoritmo con:

- **Pseudocódigo** o el **Lenguaje C de Programación** debemos utilizar órdenes (sentencias) específicas para cada acción:

ACCIÓN	PSEUDOCÓDIGO	LENGUAJE C
Inicio del programa, es el comienzo de la estructura principal del programa	INICIO/COMIENZO	main() {
Ingreso de datos	LEER	scanf();
Salida de datos	MOSTRAR	printf();
Fin del programa (cada estructura que comienza debe terminar)	FIN	}

Si el Algoritmo lo resolvemos con **Pseudocódigo**:

- **Comienzo Programa_1**
- **Leer L1;**
- **Leer L2;**
- **Area L1 * L2;**
- **Mostrar Area;**
- **Fin**

Comentario: Todo algoritmo tiene un principio y un fin, y lo que necesite para resolverlo son las acciones que se deben realizar para su resolución.

Si el Algoritmo lo resolvemos con el **Lenguaje C de Programación**:

```
main ( ) // Comienzo de Programa_1
{
    scanf("valor", &L1);
    scanf("valor", &L2);
    Area = L1 * L2;
    printf ("valor", Area);
} // Fin Programa_1
```

Comentario: Todas las sentencias (acciones) que se utilicen para la resolución de un problema deben estar entre llaves, que nos indicaran el principio y el fin de una estructura.

Datos

Tipos y Estructuras

TIPOS DE DATOS

- Cada tipo de dato ocupará un espacio en memoria RAM y el espacio ocupado se mide en bytes y cada byte es igual a 8 bits.
  - Los valores del tipo **char** se utilizan para guardar todo tipos de caracteres, solos o como texto.
  - Los valores de tipo **int** se usan para guardar cantidades enteras.
  - Los valores del tipo **float** o **double** se usan para guardar números con decimales.

TIPOS DE DATOS

- **En la tabla se muestran los tipos de datos básicos para el Lenguaje C:**

TIPOS	TAMAÑO EN BITS	RANGO
char	**8**	0 a 255
int	**16**	-32768 a 32767
float	**32**	3.4e-38 a 3.4e+38
double	**64**	1.7e-308 a 1.7e+308

ESTRUCTURAS DE DATOS

- **ARREGLOS (ARRAYS)**
  - Los arreglos son tipos de datos que reservan un espacio en la memoria RAM para almacenar temporalmente los datos ingresados.
  - La cantidad de espacio que se reserve dependerá del tipo de dato y de la cantidad de elementos que lo compongan.
  - Los arreglos pueden almacenar tanto caracteres como cadena de caracteres y valores numéricos de distinto tipo (enteros y con decimales).

TIPOS DE ARREGLOS (ARRAYS)

- **VECTORES**
  - Que son arreglos unidimensionales.
- **MATRICES**
  - Son arreglos bidimensionales o multidimensionales.

VARIABLES Y CONSTANTES

VARIABLES

- **¿Qué son?**
  - Son objetos de un programa cuyo valor puede cambiar durante la ejecución del mismo.
- **¿Para que se utilizan?**
  - Para almacenar los distintos tipos de datos dentro de un programa.
- **¿De que tipo existen?**
  - Globales
  - Locales

Cómo se declaran las Variables:

En Pseudocódigo	En C
entero cantidad;	int cantidad;
real valor;	float valor;
caracter letra;	char letra;

- Para poder usar una variable debe declararse.

Como se inicializan las Variables:

En Pseudocódigo	En C
entero cantidad 10;	int cantidad = 10;
real valor 8.2;	float valor = 8.2;
caracter letra ‘a’ ;	char letra = ’a’;

- Si necesitamos darle un valor inicial a una variable debe inicializarse.

Como se declaran las estructuras de datos:

- **Ejemplos de arreglos de caracteres (**Cada carácter ocupa en memoria 1 byte)**:**
  - **VECTORES:**
    - **char palabra[15];**
  - **MATRICES:**
    - **char nombres[5][20];**

Como se declaran las estructuras de datos:

- **Ejemplos de arreglos numéricos (**El tamaño que ocupen dependerá del tipo de datos)**:**
  - **VECTORES:**
    - **int edades[20];**
  - **MATRICES:**
    - **float precios[10][5];**

Como se inicializan las estructuras de datos:

- **Ejemplos de arreglos de caracteres:**
  - **VECTORES:**
    - **char palabra[15]={‘H’, ‘o’, ‘l’, ‘a’, ‘\0’};**
  - **MATRICES:**
    - **char nombres[2][10]={“Ana”,“Juan”};**

Como se inicializan las estructuras de datos:

- **Ejemplos de arreglos numéricos:**
  - **VECTORES:**
    - **int edades[5]={10, 66, 55, 31, 18};**
  - **MATRICES:**
    - **float precios[1][3]={5.5, 9.99, 10.50};**

CONSTANTES

- **¿Qué son?**
  - Es un dato invariable a lo largo del programa, es un valor que no puede cambiar durante su ejecución.
- **¿De que tipo existen?**
  - Las constantes se pueden utilizar para cualquier tipo de dato.

Como se declaran las Constantes:

En Pseudocódigo	En C
MAXIMO 100	**#define** MAXIMO 100
NOMBRE “LA FACULTAD”	**#define** NOMBRE “LA FACULTAD”
SIMBOLO ‘a’	**#define** SIMBOLO ‘a’
VALOR 7.50	**#define** VALOR 7.50

OPERADORES Y OPERANDOS

TIPOS DE OPERADORES

DEFINICIONES:

- **Operadores:**
  - Son elementos que relacionan de forma diferente los valores de una o más variables y/o constantes. Es decir, los operadores nos permiten manipular valores.
- **Operandos:**
  - Son los datos a ser procesados.

TIPOS DE OPERADORES:

- **Asignación**
- **Aritméticos**
- **Relacionales**
- **Lógicos**

CLASIFICACIÓN:

- La asignación se puede clasificar de la siguiente forma:
  - **Simples**
  - **Contador**
  - **Acumulador**

CLASIFICACIÓN:

- **Simples:**
  - Consiste en pasar un valor constante a una variable (el valor de la derecha se le asigna al valor de la izquierda):

En Pseudocódigo	Ejemplo	En C	Ejemplo
□	a □ 15	=	a = 15

CLASIFICACIÓN:

- **Contador:**
  - Es una variable que se incrementa o decrementa en una unidad o en una cantidad constante:

En Pseudocódigo	En C	Acción
contador ← contador + 1	contador = contador + 1	Incrementa
contador ← contador - 1	contador = contador - 1	Decrementa

Comentario: Los contadores se utilizan generalmente para contar la cantidad de veces que necesitamos que se repita una acción dentro de un programa.

CLASIFICACIÓN:

- **Acumulador:**
  - Es una variable que se incrementa en una cantidad variable:

En Pseudocódigo	En C	Acción
suma ← suma + numero	**suma = suma + numero**	**Acumulador**

Comentario: Los acumuladores se utilizan para calcular la suma total de los valores ingresados durante la ejecución de un programa. Un acumulador es igual a una sumador.

OPERADORES ARITMÉTICOS

En Pseudocódigo	Ejemplo	En C	Ejemplo	Acción
-	a 6 - 3	**-**	a = 6 - 3	Resta
+	a 6 + 3	**+**	a = 6 + 3	Suma
*****	a 6 * 3	*****	a = 6 * 3	Multiplicación (Producto)
/	a 6 / 3	**/**	a = 6 / 3	División
Mod	a 6 mod 3	**%**	a = 6 % 3	Resto de la división
Fórmula	a a + 1	**++**	a = a + 1 a++	Incremento
Fórmula	a a - 1	**--**	a = a - 1 a--	Decremento

OPERADORES RELACIONALES

En Pseudocódico	En C	Acción
>	>	Mayor que
<	<	Menor que
>=	>=	Mayor o igual que
<=	<=	Menor o igual que
=	==	Igual que
<>	!=	Distinto que o no igual que

OPERADORES LÓGICOS

En Pseudocódigo	En C	Acción
And / Y	**&&**	Producto Lógico
Or / O	**||**	Suma Lógica
Not / No	**!**	Negación

TABLAS DE VERDAD

AND – OR – NOT

&& - || - !

Operador AND

Operando 1	Operando 2	Resultado
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

- **Producto Lógico:**
  - Si uno y sólo uno de los valores de entrada en falso, el resultado es falso.
    - Por ejemplo:
      - 0 * 0 = **0**
      - 0 * 1 = **0**
      - 1 * 0 = **0**
      - 1 * 1 = **1**

Operador OR

Operando 1	Operando 2	Resultado
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

- **Suma Lógica:**
  - Si uno de los valores de entrada es verdadero, el resultado es verdadero.
    - Por ejemplo:
      - 0 + 0 = **0**
      - 0 + 1 = **1**
      - 1 + 0 = **1**
      - 1 + 1 = **1**

Operador NOT

Operando	Resultado
0	**1**
1	**0**

- **Negación:**
  - Nos devuelve como resultado la inversa del valor ingresado.
    - Por ejemplo:
      - 0 = **1**
      - 1 = **0**