PROGRAMACIÓN I

Unidad 1

Composición Secuencial de Acciones. Estados iniciales y finales. Estados intermedios y Refinamiento sucesivos. La estructura de decisión. Estructura de decisión múltiple. Tipos de estructura de Control: Sentencia IF-Else. Sentencia Switch. Sentencias anidadas. Principio de Inducción. La estructura de Iteración. Tipos de estructura de iteración: Sentencia While. Sentencia For. Sentencia Do-While. Programación Esquemática.

Programación Estructurada

Esta forma de programar (paradigma) se basa en un famoso teorema, desarrollado por Edsger Dikjstra, que demuestra que todo programa puede escribirse utilizando únicamente las tres estructuras básicas de control:

- **Secuencia: el bloque secuencial de instrucciones, ejecutadas sucesivamente, una detrás de otra.**
- **Selección: la instrucción condicional con doble alternativa, de la forma "*SI* condición *then* instrucción-1 *sino* instrucción 2".**
- **Iteración: el bucle condicional "while condición do instrucción", que ejecuta la instrucción repetidamente mientras la condición se cumpla.**

Sentencias de Control

Las sentencias de control permiten controlar el flujo del programa, tomando decisiones a partir de comparaciones.

- **Se usan instrucciones condicionales y de ciclos.**
- **Un condicional es un conjunto de sentencias que pueden o no ejecutarse, dependiendo del resultado de una condición.**
- **Un ciclo es un conjunto de sentencias que son ejecutadas varias veces, hasta que una condición de término es satisfecha.**
- **Tanto los condicionales como los ciclos contienen a otras sentencias. Para indicar esta relación ,las sentencias contenidas no se escriben en la misma columna que la sentencia de control, sino un poco más a la derecha**

Sentencias de Control

- Las instrucciones **condicionales** son:
  - SI
  - SWITCH
- Las instrucciones de **ciclo** son:
  - WHILE
  - FOR
  - REPEAT

Operadores Relacionales

Se usan para expresar condiciones y describir una relación entre dos valores

	OPERADOR	DESCRIPCIÓN
BINARIOS	>	Mayor que
	>=	Mayor o igual que
	<	Menor que
	<=	Menor o igual que
	==	Igual que
	!=	DSIerente que

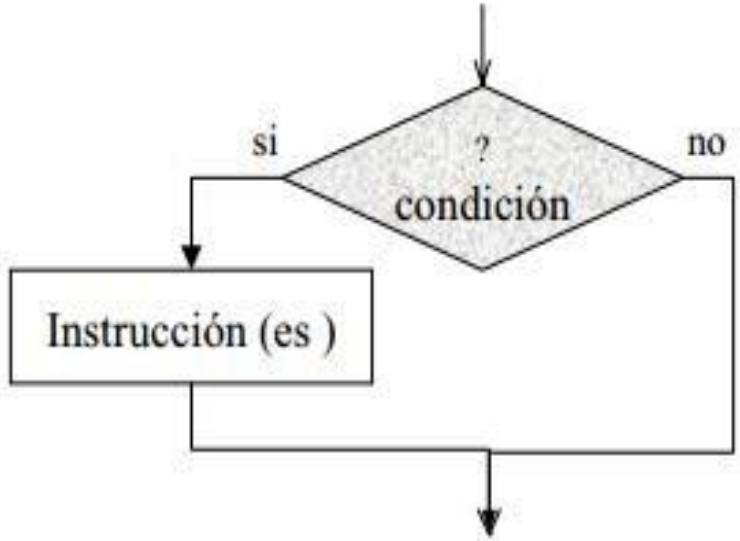
Sentencia IF

Sentencia SI simple

La estructura si adopta una de las dos formas siguientes:

Si CONDICIÓN entonces ACCION;
Fin si

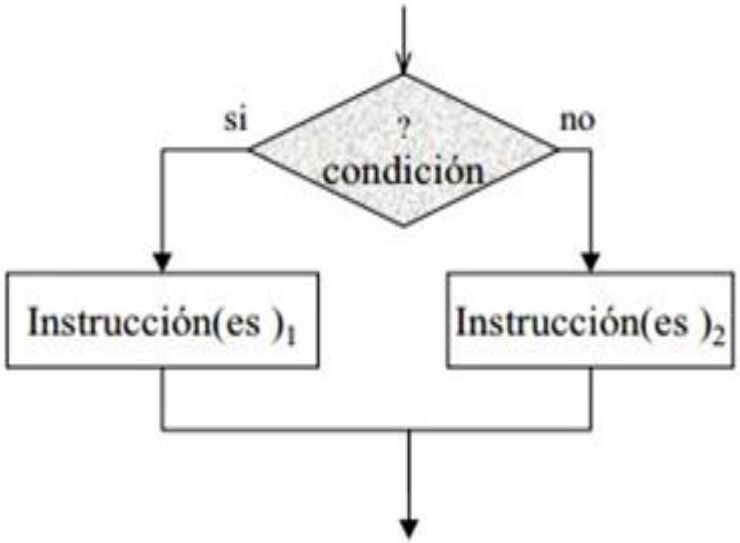
en donde condición es una sentencia que se evalúa como verdadera



Sentencia SI doble

Si CONDICIÓN entonces ACCIÓN 1;
sino ACCIÓN 2;
Fin si

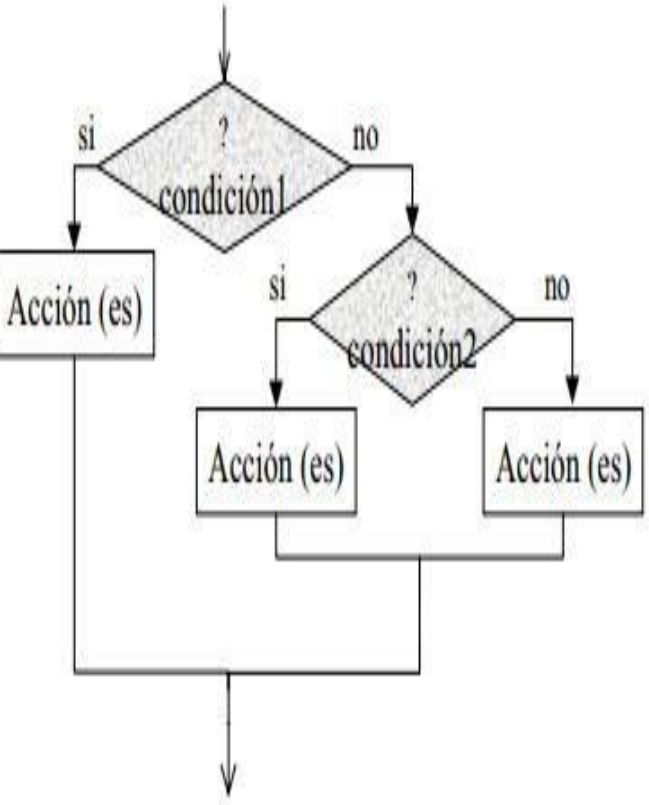
en donde *expresión* es una sentencia que se evalúa como verdadera (devuelve un valor no nulo) o falsa (devuelve cero). La palabra *sentencia* puede ser una sentencia simple terminada o un grupo de sentencias .



Sentencia SI anidada

- **Es posible utilizar las instrucciones SI-SINO anidadas, es decir, que alguna de las ramas sea a su vez otra instrucción SI-SINO.**

- **Permite implementar decisiones que implican más de dos alternativas.**



Si CONDICIÓN entonces ACCIÓN 1;
sino
Si CONDICIÓN entonces ACCIÓN 2;
sino
……
Si CONDICIÓN entonces ACCIÓN n-1;
sino ACCIÓN n;
Fin si

Operadores Lógicos

Actúan sobre expresiones booleanas, es decir, sobre valores *verdadero* o *falso* generados por expresiones como las explicadas en el caso anterior.

	OPERADOR	***DESCRIPCIÓN***
UNARIOS	**-**	*not*
BINARIOS	**∧**	*Y (and)*
	∨	*O (or)*

El resultado de una operación lógica viene dado por su tabla de verdad

Cuando se ejecuta un programa, se pueden producir tres tipos de errores:

1. *Errores de compilación*. Se producen normalmente por un uso incorrecto de las reglas del lenguaje de pr gramación y suelen ser *errores de sintaxis*. Si existe un error de sintaxis, la computadora no puede compre der la instrucción, no se obtendrá el programa objeto y el compilador imprimirá una lista de todos los error encontrados durante la compilación.

2. *Errores de ejecución*. Estos errores se producen por instrucciones que la computadora puede comprender pero no ejecutar. Ejemplos típicos son: división por cero y raíces cuadradas de números negativos. En estos casos se detiene la ejecución del programa y se imprime un mensaje de error.

3. *Errores lógicos*. Se producen en la lógica del programa y la fuente del error suele ser el diseño del algoritmo. Estos errores son los más difíciles de detectar, ya que el programa puede funcionar y no producir errores de compilación ni de ejecución, y sólo puede advertirse el error por la obtención de resultados incorrectos. En este caso se debe volver a la fase de diseño del algoritmo, modificar el algoritmo, cambiar el programa fuente y compilar y ejecutar una vez más.

```
Acción Compara es
 Ambiente
 hora:entero;

 Algoritmo

 Leer ( hora);

  SI ((hora >= 0) ∧ (hora < 12)) escribir( "Buenos días" );
    sino
        SI ((hora >= 12) ∧ (hora < 18)) escribir( "Buenas tardes" );
           sino
                SI ((hora >= 18) ∧ (hora < 24)) escribir( "Buenas noches" );
                     sino escribir( "Hora no válida" );
                 fin si
          fin si
   fin si

 Fin Acción
  }
```

Consigna

Calcular el promedio de tres valores que se ingresan por teclado

```
Acción Suma es
Ambiente
        n1,n2,n3:entero;
        prom:real;

Algoritmo
                Escribir: (“Ingrese n1”);
                Leer (n1);
                Escribir: (“Ingrese n2”);
                Leer (n2);
                Escribir: (“Ingrese n3”);
                Leer (n3);

                Si ((n1>0) ∧(n2>0) ∧(n3>0))  entonces
                                                prom:=(n1+n2+n3)/3;
                                                Escribir(prom);
                        sino Escribir(“Error”);

facción
```

Consigna

Ejercicio: Calcular la división entera de dos números que se ingresan por teclado.

Consigna

Ejercicio: Ingresar dos caracteres por teclado. Si son distintos y uno es distinto de a mostrar por pantalla su nombre y en caso contrario mostrar su apellido.