Programación: Programación estructurada

En esta Unidad contamos con una guía turística de lujo: Ada Lovelace. Es experta en automatizar procesos mediante una serie de instrucciones escritas en un lenguaje de programación. A ella se le atribuye el concepto de "bucle", que aprenderemos a continuación y es considerada la primera programadora de la historia. Pero ¿qué significa programar?

Actividad de lectura

El diccionario de la Real Academia Española (R.A.E.) define el verbo ***programar*** así:

- *Idear y ordenar las acciones necesarias para realizar un proyecto.*
- *Preparar ciertas máquinas por anticipado para que empiecen a funcionar en el momento previsto.*
- *Preparar los datos previos indispensables para obtener la solución de un problema mediante una calculadora electrónica.*

En las tres acepciones aparecen los elementos básicos de la programación estructurada. ¿Serías capaz de deducirlos?

1.Definición

Importante

La **programación estructurada** es una técnica para la mejora de la productividad en programación basada en el uso de bloques o secuencias de instrucciones organizadas e interrelacionadas de tal forma que es posible leer su codificación de principio a fin de forma continua.

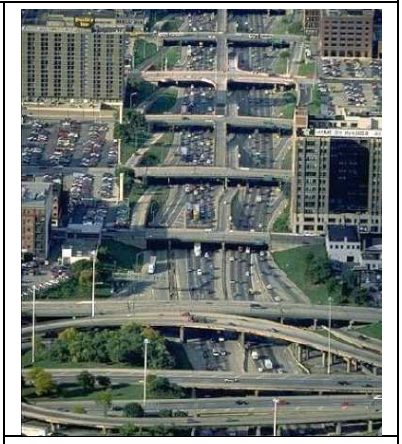
Img 1. Programación no estructurada

Imagen de Diariomotor con licencia CC

Este tipo de programación exige al programador el trabajo con las mínimas bifurcaciones o desviaciones de control de la estructura principal. Su objetivo es la mejora de la eficiencia y la fiabilidad tanto en la fase de pruebas como en la modificación y mantenimiento del programa.

El resultado es un programa fácilmente comprensible, independientemente de la lógica del programador.

Los programas están formados por bloques, que a su vez pueden contener pequeños subprogramas. Por tanto, su prueba y compilación se realiza por segmentos independientes, ahorrando tiempo y facilitando la localización de errores.



Img 2. Programación estructurada

Imagen de Diariomotor con licencia CC

Autoevaluación

a) Un programa estructurado debe poder leerse de principio a fin.

Verdadero ○ Falso ○

b) Tanto el programa en sí como cada uno de lo bloques que lo forman tiene unas entradas y unas salidas definidas.

Verdadero ○ Falso ○

c) Un programa estructurado puede tener bifurcaciones en los bloques que lleven a otros subprogramas.

Verdadero ○ Falso ○

d) Cómo los bloques están relacionados, cuando se realizan las pruebas del programa, es necesario compilarlo de principio a fin.

Verdadero ○ Falso ○

Las ventajas de la programación estructurada son:

1. facilidad de lectura y comprensión.- el programa puede leerse de principio a fin, no contiene saltos ni bifurcaciones en su lógica y su estructura es clara.
2. facilidad de prueba.- la localización de errores es rápida, puesto que el seguimiento del programa es secuencial y, si el programa está constituido por bloques, se realiza compilando cada bloque por separado.
3. reducción de los costos de mantenimiento y facilidad de optimización.
4. mejora de la presentación del programa y de la documentación.

Autoevaluación

Completa los espacios en blanco con las palabras que aparecen en la parte inferior.

Los programas estructurados tienen una estructura [ ] que permite su [ ] de principio a fin. Su facilidad de [ ] se debe a la fácil localización de los errores.

Los programas son sencillos reduciéndose así los costos de [ ] y facilitando su [ ] .

Aunque la estructura secuencial puede hacer que los programas sean más largos, en general, su presentación y su [ ] son claras y sencillas.

documentación	optimización	mantenimiento	prueba	lectura	clara

Comprobar

Para saber más

La presentación es una de las claves en los programas estructurados. La secuencialidad facilita el seguimiento del programa y su lectura por parte de cualquier programador. La división en subprogramas y bloques claramente diferenciados también facilita la lectura.

La escritura de las instrucciones en diferentes márgenes a la izquierda es una técnica utilizada en programación para organizar las instrucciones.

Esta técnica se denomina **indentación** (del inglés *indentation*). Es castellano se denomina sangría.

La indentación se rige por normas distintas según el lenguaje de programación, aunque no suele ser problemática porque los compiladores no procesan los espacios en blanco.

Para saber más sobre este término puedes consultar estas páginas web:

- wikipedia_indentación
- indentación_del_código_fuente

```php
<?php

if($start){
    $var1 = "variable1";
    $var2 = "variable2";

    conectarBBDD();

    if($i == 23){
        $var3 = "variable3";

        while($var3 != 1){
            echo $var3;
            echo $var1.$var2;
        }
    }

    exit;
}

?>
```

Img 3. Ejemplo de sangrías en código PHP

Imagen obtenida de PHPnight.com

1.1.Teorema de la programación estructurada

La definición de programa se basa en el cumplimiento de las siguientes condiciones:

1. Existe una entrada y una salida.
2. Existen líneas o caminos desde la entrada hasta la salida que llevan al siguiente bloque del programa. Es decir, no existen bloques infinitos ni instrucciones que no se ejecutan.

El teorema del programa estructurado completa esta definición acotando las estructuras lógicas de control que pueden usarse.

Importante

El teorema del programa estructurado dice que toda función computable puede ser implementada en un lenguaje de programación que combine tres subrutinas de control: secuencia, selección e iteración.

El teorema dice que las instrucciones de carácter incondicional no son necesarias. Por tanto, siempre habrá un modo de sustituir los GOTO, EXIT y RETURN por otras estructuras de control.

Estas tres instrucciones, denominadas de transferencia incondicional, derivan el flujo vertical de programa, que debe poder seguirse de arriba a abajo.

La filosofía de la programación estructurada es que un programa se divide en bloques con una entrada y una salida que se ejecutan secuencialmente uno detrás de otro.

Img 4. Bloques

Imagen de kinuma.com

Para saber más

Este teorema no es tan actual como podría parecer.

Si la evolución de las TIC ha sufrido su gran revolución en los 80 y los 90 en cuanto a hardware, aplicaciones y telecomunicaciones, la evolución en cuanto a programación es mucho anterior.

El nacimiento de la informática se establece con la aparición de máquinas capaces de interpretar y ejecutar secuencias de instrucciones, es decir, programas informáticos.

El teorema fue enunciado en 1960 por Böhm y Jacopini, aunque sus antecedentes se encuentran mucho antes, en la arquitectura de Von Neumann y los estudios del matemático Kleene de 1946.

En los primeros albores de la informática programar de forma eficaz era una necesidad, más que un mero avance.

Puedes encontrar más información en: wikipedia_teorema_programa_estructurado.

Autoevaluación

a) El teorema es válido para todos los tipos de programas.

- ○ Verdadero
- ○ Falso

b) ¿Cuál de estas características es propia de un programa estructurado?

- ○ Se puede leer por bloques.
- ○ Se puede leer de principio a fin de forma continua.

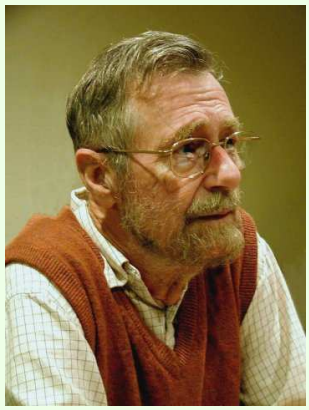
Curiosidad

ELIMINACIÓN DE LA SENTENCIA GOTO

La programación no estructurada se basa en la ejecución de bloques de sentencias o procedimientos de la misma forma que la programación estructurada. La diferencia estriba en que un programa estructurado puede leerse de principio a fin secuencialmente, mientras que un programa no estructurado permite derivar el flujo a otros bloques saltando la secuencia lógica del programa.

El uso de las estructuras básicas de control hace que la eliminación de la sentencia GOTO no sea complicado, facilitando así la lectura del programa. Sin embargo existen casos en que su utilización podría estar justificada (por ejemplo, ejecución de un programa en varios idiomas).

El rechazo al uso de esta instrucción fue plasmado por el físico holandés Dijkstra en 1968 en su artículo "Instrucción Go To considerada dañina" ("Go To Statement Considered Harmful"). Puedes consultar el contenido de este artículo en wikipedia: GOTO



Img 5. Dijkstra

1.2. Estructuras lógicas de control

Importante

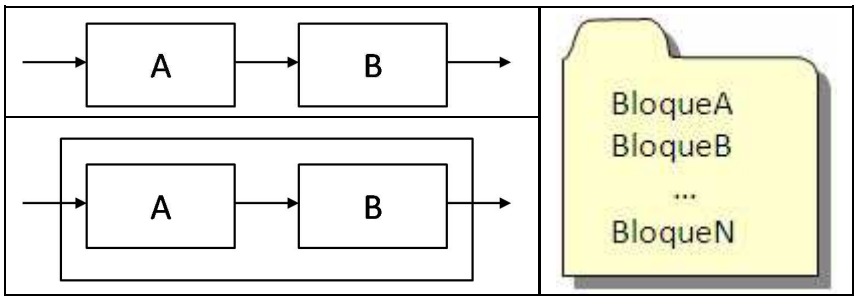
Las **estructuras lógicas de control** son:

- **Secuencia**.- sucesión lineal de instrucciones o subrutinas.
- **Selección**.- bifurcación condicional para ejecutar una u otra subrutina.
- **Iteración**.- repetición de una instrucción o subrutina mientras se cumple una condición.

Secuencia

Los bloques o subrutinas que componen el programa se ejecutan en el orden en el que aparecen. Estos bloques pueden ser simples instrucciones o programas en sí mismos (tienen una entrada y una salida definidas y ejecutan una tarea).

La unión de varios bloques compone a su vez un programa estructurado.



Ejercicio resuelto

Problema: dado el radio de un círculo, calcular su área y su longitud.

El programa deberá realizar las siguientes acciones:

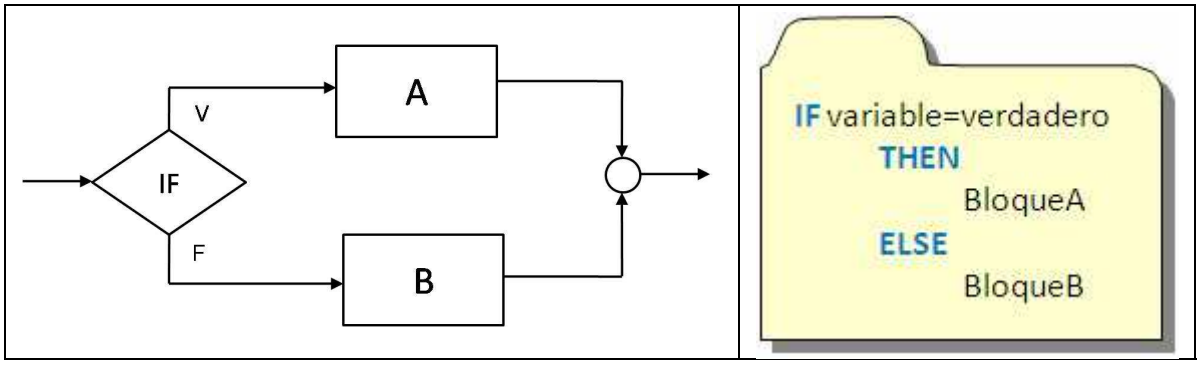
1. Leer el valor del radio tecleado.
2. Asignarlo a la variable radio.
3. Calcular el área multiplicando la constante pi por el radio al cuadrado.
4. Mostrar el resultado de la variable área.

Selección

Un bloque se ejecuta si una condición es verdadera. Se basa en la elección entre dos opciones y se denomina IF-THEN-ELSE (Si-entonces-si no). En esta estructura, la condición debe poder tomar únicamente los valores verdadero y falso, es decir debe ser booleana.

La pregunta se representa con la figura de un rombo cuyas salidas son las entradas de los bloques A y B.

Si la condición es verdadera, se ejecutará el bloque A. Si la condición es falsa, se ejecutará el bloque B. En ambos casos, ofrece una salida.



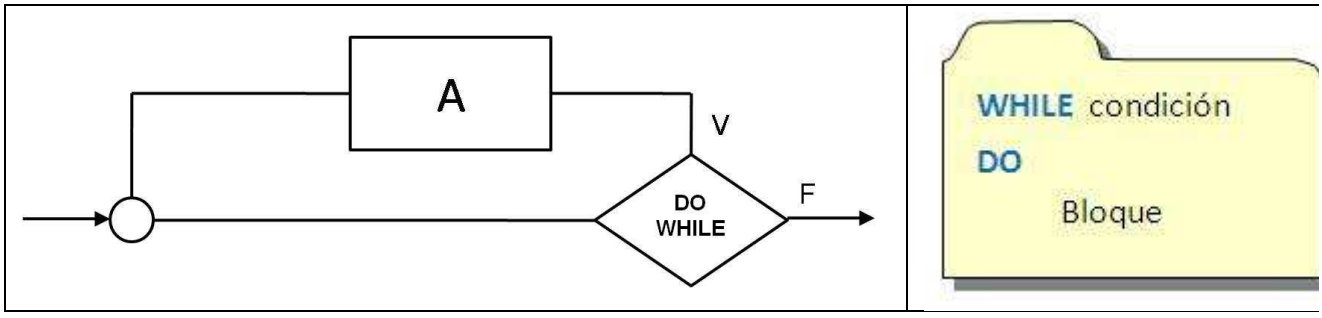
Ejercicio resuelto

Problema: comparar dos números distintos y decir cuál es mayor.

1. Leer los dos números y asignarlos a las variables numeroA y numeroB.
2. Comparar ambas variables.
3. Si numeroA es mayor que numeroB escribir "A es mayor que B".
4. Si no, escribir "B es mayor que A".

Iteración

La acción o subrutina se ejecutará mientras el valor de una condición sea verdadero. Se denomina DO-WHILE (hacer-mientras).



Ejercicio resuelto

Problema: calcular el factorial de un número entero.

1. Leer el número entero y asignarlo a la variable contador.
2. Asignar a la variable factorial el valor 1 para comenzar el proceso.
3. Mientras la variable contador sea mayor que 0, multiplicar el factorial por el contador.
4. Restar 1 a la variable contador.

Autoevaluación

¿Qué estructura utilizarías para realizar las siguientes operaciones?

1.- Programar el secundero de un reloj.			
a) Secuencia	b) Selección	c) Iteración	☐
2.- Imprimir un texto introducido mediante el teclado y mostrarlo en pantalla.			
a) Secuencia	b) Selección	c) Iteración	☐
3.- Hacer un censo por sexos.			
a) Secuencia	b) Selección	c) Iteración	☐

Comprobar

1.3. Otras estructuras lógicas

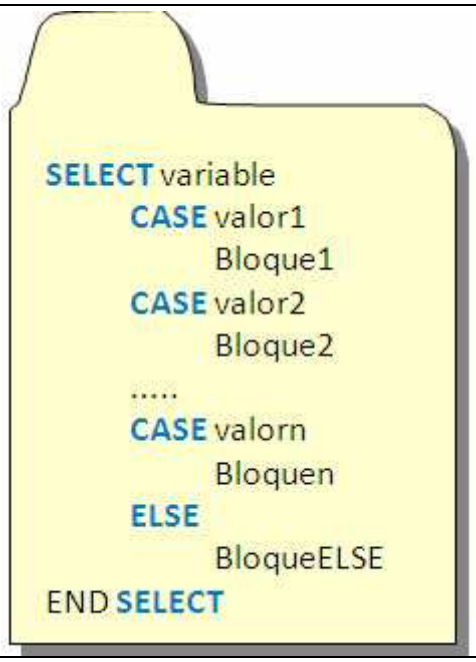
Aunque todas los programas estructurados pueden ser escritos con las tres estructuras básicas de control, existen otras estructuras que pueden ser útiles en casos particulares para simplificar el código y la lógica del programa.

Importante

Las estructuras lógicas de control SELECT-CASE, DO-UNTIL Y FOR NEXT pueden utilizarse para simplificar el código de un programa estructurado siempre que se cumplan los principios del teorema de la programación estructurada.

SELECT-CASE

Se usa cuando la variable evaluada puede tomar más de dos valores, evitando así utilizar varios IF-THEN-ELSE encadenados.

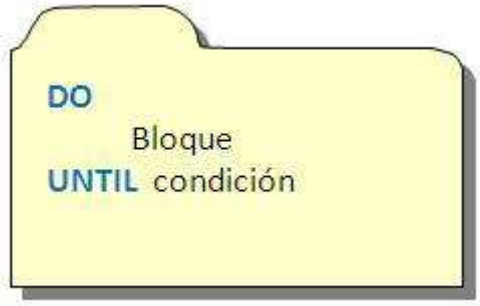


El proceso seguido consiste en:

- Se pregunta a la variable a evaluar (SELECT).
- Se busca el CASE para el valor de la variable y se ejecuta el bloque correspondiente.
- Si no coincide ningún CASE se ejecuta la sentencia ELSE.

DO-UNTIL

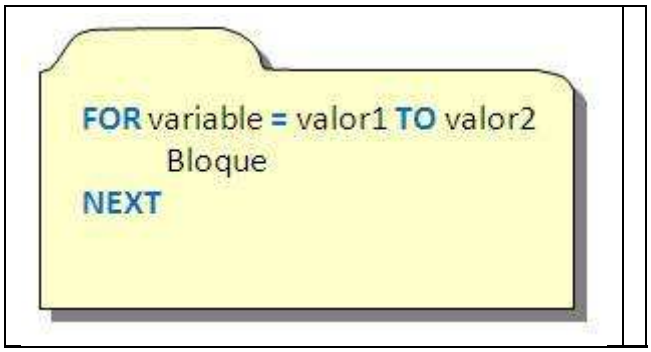
Esta estructura forma parte de las estructuras de iteración. La diferencia con DO-WHILE es que el bloque o subrutina se ejecutará hasta que una condición sea verdadera. En esta estructura el bloque se ejecuta por lo menos una vez, en cambio en el DO-WHILE puede que no se llegue a ejecutar nunca.

	La pregunta se realiza al final de la estructura.

FOR-NEXT

Otras de las estructuras de iteración es el bucle FOR-NEXT. En esta estructura el bloque o subrutina se ejecutará mientras la variable recorra los valores comprendidos entre el valor1 y el valor2. Por ello, se denomina estructura de recorrido.

Se puede determinar el incremento que se aplicará a la variable en cada ciclo del bucle con el término STEP.



Autoevaluación

Completa los espacios en blanco con palabras que aparecen en la parte inferior.

- Cualquiera de estas tres estructuras de [ ] pueden ser sustituidas por una de las tres estructuras [ ] .
- El bloque de sentencias que contiene la estructura DO-UNTIL se ejecuta [ ] que la condición sea verdadera. Esta estructura pertenece a las de [ ] .
- La estructura SELECT-CASE puede ejecutar tantas [ ] distintas como valores pueda tomar la variable. Esta estructura pertenece a las de [ ] .
- En la estructura FOR-NEXT la subrutina se repetirá tantas veces como valores tome la [ ] , por lo que se denomina [ ] o de recorrido.

variable	iteración	selección	básicas	bucle	subrutinas	control	hasta

Comprobar

Para saber más

BUCLE

En este apartado ha surgido el término *bucle*, referido a un grupo de instrucciones que se repite un número de veces determinado por el valor de una variable. Su función primordial es el ahorro de código cuando una subrutina debe repetirse para varios valores, evitando así la sentencia GOTO para iterar en un programa.

Cuando la condición de finalización del bucle no se cumple en ningún caso, se dice que es un *bucle infinito* porque se repite de forma indefinida y el programa no puede continuar. Un ejemplo sería un bucle en el que la variable sólo tome valores pares y la condición de salida sea variable=5. Por ello, la definición dice que para todo bucle deben definirse condiciones de inicio y finalización de éste. Un bucle infinito suele considerarse un error en el programa, salvo si ha sido diseñado así por el programador (programas maliciosos).

Img 6. Biblioteca Bucle infinito de Job Koelenwinj

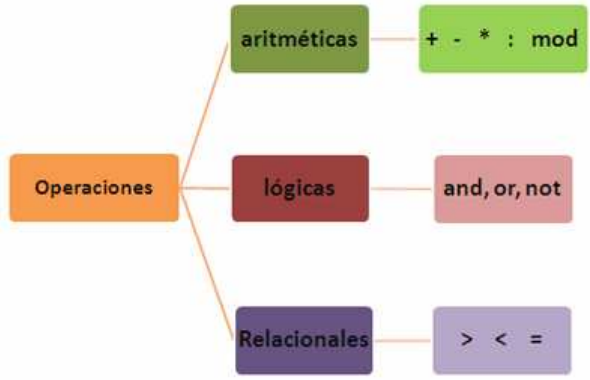
Imagen de Blogodisea

2. Elementos

En el tema anterior has aprendido que para programar se usan distintos lenguajes. En cada uno de ellos la instrucciones se escriben de forma distinta y las variables deben cumplir unas condiciones.

La programación estructurada se rige por unas normas comunes a todos los lenguajes en cuanto a la declaración de variables y a la redacción de los procedimientos.

Las operaciones que se realizan con dichas variables pueden ser aritméticas, lógicas y relacionales. Estas operaciones se suelen agrupar para realizar tareas concretas en bloques de instrucciones denominados procedimientos.



2.1. Variables y constantes

Los datos de entrada, de proceso y de salida de un programa se guardan en variables.

Importante

Una **variable** es un espacio de memoria, con un nombre especificado por el programador, que permite almacenar datos durante la ejecución de un programa.

La primera tarea será declarar el nombre de las variables que se van a utilizar en un programa para que éste pueda reconocerlas y asignarlas el espacio de memoria necesario.
El valor de las variables puede variar durante la ejecución del programa. Si su valor no varía se dice que es una constante, pero realmente es una variable con un valor fijo en ese programa. (un ejemplo es asignar a la variable pi el valor 3,14).
Los valores que puede tomar una variable deben ser establecidos previamente. Estos valores pueden determinar el contenido de la variable o el rango válido para ese programa. Según su contenido, las variables se clasifican en:

Numéricas	Almacenan números positivos o negativos, enteros o con decimales.
Alfanuméricas	Almacenan cadenas de caracteres que pueden contener letras, números y caracteres especiales.
Lógicas	Son variables booleanas que solamente pueden tomar los valores verdadero o falso.

Autoevaluación

¿En qué tipo de variable almacenarías cada dato?

A: alfanumérica	B: Booleana	C: constante	N: numérica

Casado	☐	factorial	☐
IVA	☐	apellidos	☐

Comprobar

Las variables se usan para almacenar datos que pueden ser introducidos por el usuario al ejecutar el programa o utilizarse para guardar temporalmente el resultado de las operaciones.

Según su uso, las variables se clasifican en:

de trabajo	son las variables destinadas a almacenar el resultado de las operaciones matemáticas y los datos durante la ejecución del programa.
contadores	almacenan el número de veces que se realizará un ciclo o que se ha realizado. La cuenta puede ser hacia adelante o hacia atrás, incrementando o disminuyendo su valor en cada ciclo.
acumuladores	son variables de trabajo destinadas a almacenar el resultado de operaciones repetitivas cuyo valor se va acumulando en la variable.

Autoevaluación

En el apartado 1.2. del tema has encontrado distintas variables en los programas de ejemplo. Di que uso tiene cada una de ellas.

radio		numeroA	
factorial		contador	

Comprobar

Declaración de variables

Antes de comenzar a trabajar es necesario describir las variables que se van a utilizar. Esta operación se denomina declaración de variables.

Importante

La **declaración de variables** consiste en la asignación de nombres y tipo de contenido al principio del algoritmo.

Las variables numéricas pueden ser de tipo entero, positivo o negativo, real, ...

Las variables alfanúmericas pueden ser un sólo carácter o una cadena de caracteres.

Aunque no todos los lenguajes de programación requieren que se declaren las variables, suele ayudar a la comprensión del programa.

Autoevaluación

En estos programas puedes observar la declaración de variables en diferentes lenguajes de programación.

Pseudocódigo:

```
INICIO
    Base, Altura: ENTERO
    ESCRIBA "Diga la Base: "
    LEA Base
    ESCRIBA "Diga la Altura"
    LEA Altura
    ESCRIBA "Area del Triangulo = ",
            (BASE*ALTURA)/2
FIN
```

Pseudocódigo:

```
INICIO
  Not1, Not2, Not 3 :REAL
  Def: REAL
  LEA Nota1, Nota2, Nota3
  Def ß (Not1 + Not2 + Not3) /3
  Si Def < 3 entonces
    Escriba "Reprobó el curso"
  Sino
    Escriba "Aprobó el curso"
  Fin-Si
FIN
```

```
var

    I, J, K     : Integer;  (* valores enteros *)
    Contador    : Integer;
    Radio       : Real;     (* valor real  *)
    Letra       : Char;     (* un caracter *)
```

Curiosidad

Aunque en estos fragmentos de código no aparece, existe un tipo de variable denominado **float**.

¿A qué tipo de datos se refiere?

Puedes encontrar información en estás direcciones web: wikipedia_float, variables_en_java

Para saber más

Las variables pueden ser de ámbito local o global dependiendo de si pueden ser utilizadas (leídas o modificadas) solamente en el procedimiento en el que se declaran o en todos los procedimientos o subrutinas del programa, respectivamente. Cuando las variables son accesibles incluso desde varios programas se denominan superglobales.

En wikipedia puedes encontrar un ejemplo del ámbito de una variable en lenguaje Java.

2.2. Operaciones

El objetivo de un programa es realizar operaciones con los datos que almacenan las variables.

Importante

Una **operación** es una acción sobre una variable que ofrece un dato de salida.

Las operaciones que pueden realizarse se resumen en la siguiente tabla:

Aritméticas	Operaciones matemáticas con datos numéricos.	suma (+), resta (-), multiplicación (*), división (/), exponenciación (^), módulo (mod), resto.
Relacionales	Realizan comparaciones entre dos datos del mismo tipo devolviendo un valor de verdadero o falso.	mayor que (>), menor que (<), igual que (=), distinto que (<>).
Lógicas	Realizan operaciones lógicas con valores booleanos (verdadero o falso).	y (AND), o (OR), negación (NOT).

Autoevaluación

Consulta las tablas de verdad de las funciones lógicas y completa las soluciones.

¡Ah!, ¿no sabes que es una tabla de verdad? Es la representación de todas las combinaciones posibles que se pueden dar en una función lógica y sus resultados.

Recuerda que los resultados sólo pueden ser V ó F.

NOT V	☐
V **OR** F	☐
V **AND** F	☐

Comprobar

Para saber más

Prioridad de los operadores

Las reglas en cuanto a la prioridad de las operaciones son las aplicables en matemáticas en los operadores aritméticos. Los operadores lógicos tienen menor prioridad que los aritméticos. El orden será el siguiente:

1. ( )
2. ^
3. *, /, mod, NOT
4. +, -, AND
5. >, <, >=, <=, <>, =, OR

Ejercicio resuelto

Calcula el valor de salida de la siguiente operación para los valores de las entradas:

a=15	b=5	c=10	d=20

((a >= c) AND (a < d)) OR ((b >= c) AND (b < d))

2.3. Procedimientos y librerías

La programación estructurada se basa en la creación de algoritmos fáciles de leer de principio a fin, sin saltos ni derivaciones en el flujo del programa. El uso de las tres estructuras básicas de control puede hacer que el código de los programas estructurados sea extenso, lo cual, puede dificultar su lectura y seguimiento.

Para evitarlo se recurre a la segmentación, es decir, a la división del programa en bloques de código. De esta forma, los programas se dividen en procedimientos que son "llamados" desde el código del programa.

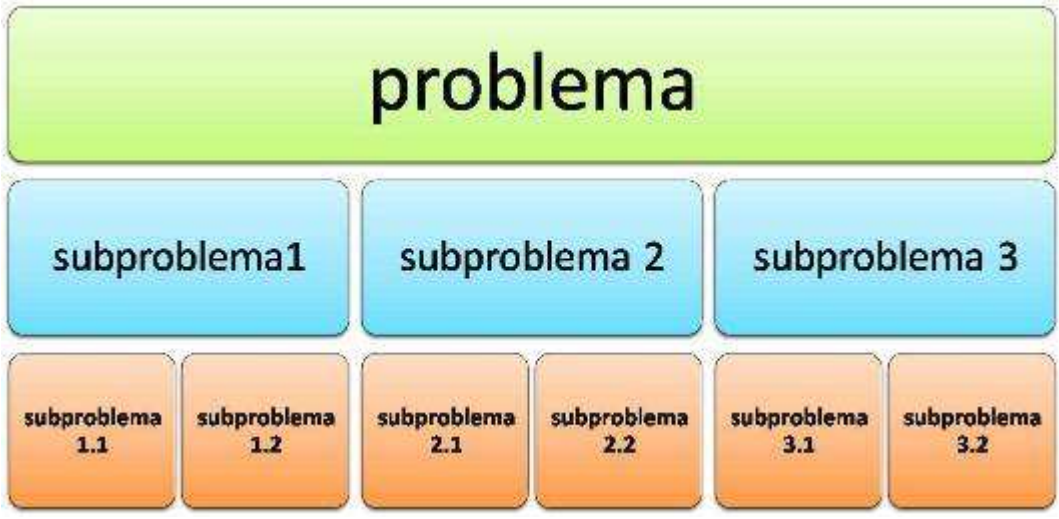
Para saber más

Top-down

El diseño de los programas estructurados se basa en la división del programa en subprogramas más sencillos. La estructura principal estará dividida en niveles de mayor a menor complejidad, interrelacionados entre sí mediante una estructura jerárquica.

Esta técnica se denomina top-down o "divide y vencerás".

Esta forma de programar se considera una mejora de la programación estructurada. Se denomina programación modular, programación por procedimientos o programación funcional y se basa en los módulos o subprogramas denominados procedimientos.



Importante

Un **procedimiento** es un conjunto de instrucciones que realiza una o varias tareas específicas. Sus componentes son:

- una entrada o entradas determinadas.
- un dato de salida.
- un nombre único e identificable en el algoritmo del programa principal.

El trabajo con procedimientos es útil en estos casos:

- cuando una tarea debe ejecutarse varias veces en un programa.
- cuando la resolución de un problema requiere un código extenso y de lectura complicada.

Sin embargo, no todos los lenguajes de programación admiten el uso de procedimientos. Además, en cada lenguaje se designan los procedimientos de un modo distinto:

Procedimientos y funciones	Pascal
Funciones	C y C++
Subrutinas	Basic y Fortran
Secciones	Cobol

Autoevaluación

Un ejemplo típico de programa modular es el utilizado en la gestión de una cuenta bancaria. Las operaciones que se pueden realizar están claras y bien definidas (imposición, retiro, transferencia, estado de la cuenta), por tanto se puede diseñar un procedimiento para cada proceso.
¿Eres capaz de proponer otros ejemplos?

Las ventajas del uso de esta "modularidad" en la programación son:

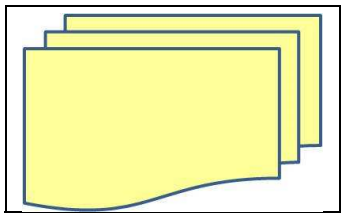
- simplificación del código.
- rapidez en el desarrollo y en la detección de errores.
- facilidad en el mantenimiento y la modificación del código.

Estas ventajas adquieren su máxima importancia cuando los procedimientos para realizar tareas típicas que se repiten en todos los programas ya están implementados y simplemente se nombran en el programa. Estos procedimientos se almacenan en una bibliotecas denominadas librerías.

Importante

Una **librería** es una colección de programas o subrutinas que pueden ser utilizadas por programas independientes, siendo llamadas mediante un programa denominado enlazador.

Las librerías no suelen ser programas ejecutables, se incorporan al código del programa, bien en la compilación o bien en la ejecución. Denominándose estáticas en el primer caso (archivos lib) y dinámicas en el segundo (archivos dll)
Todos los sistemas operativos suelen incluir una colección de librerías que implementan la mayoría de los servicios del sistema.



Curiosidad

En la definición se ha utilizado el término librería debido a que proviene del inglés *library*.
Aunque su traducción correcta es biblioteca, en el ámbito de la informática se admiten ambos términos.
Los términos que en un idioma se asemejan a otros con diferente significado se denominan *falso amigo*.

Library **» biblioteca**
Book shop **» librería**

Autoevaluación

Después de tanta información, ¿quieres comprobar todo lo que has aprendido?

a) El diseño top-down se refiere a la condición de que los programas estructurados deben leerse de principio a fin.

Verdadero ○ Falso ○

b) Un procedimiento es un conjunto de instrucciones para realizar una tarea específica.

Verdadero ○ Falso ○

c) Los procedimientos reducen el código del programa, aunque lo complican con las palabras que los identifican.

Verdadero ○ Falso ○

d) Las librerías son colecciones de pequeños programas ejecutables.

Verdadero ○ Falso ○