Índice de Contenidos

4.7.3.- Sentencias de iteración o repetitivas.

- ✓ Estructura Mientras

Prácticas resueltas

Prácticas propuestas

- ✓ Estructura Repetir

Prácticas resueltas

Prácticas propuestas

- ✓ Estructura Para

Prácticas resueltas

Prácticas propuestas

4.7.4.- Sentencias de salto.

Objetivos de la Unidad Didáctica

Enlaces de interés

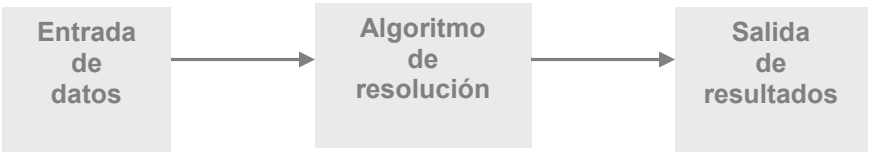
Glosario de términos

4.1.- Partes principales de un programa.

Como ya hemos comentado en el tema anterior, un programa es un conjunto de instrucciones que producirán la ejecución de una determinada tarea. Todo programa, en general contiene dos bloques bien diferenciados:

- Bloque de declaraciones.
- Bloque de instrucciones.

Dentro del bloque de instrucciones de un programa podemos diferenciar 3 partes fundamentales: entrada de datos, algoritmo de resolución y salida de resultados.



En algunos casos estas tres partes están perfectamente delimitadas, pero a veces las instrucciones quedan entremezcladas a lo largo del programa.

4.1.1.- Entrada de datos.

Lo constituyen todas aquellas instrucciones que toman datos de un dispositivo externo almacenándolos en memoria central para que puedan ser procesados.

También se consideran dentro de esta parte las instrucciones de depuración de los datos de entrada, es decir aquellos que se encargan de comprobar la corrección de los mismos.

4.1.2.- Algoritmo de resolución.

Está formado por las instrucciones que modifican los objetos a partir de un estado inicial hasta el estado final, dejando estos disponibles en la memoria central. Es decir es el proceso que transforma los datos de entrada en datos de salida.

4.1.3.- Salida de resultados.

Son el conjunto de instrucciones que toman los datos finales de la memoria central y la envían a los dispositivos externos.

4.2.- Clasificación de las instrucciones.

Las instrucciones disponibles en un lenguaje de programación dependen del tipo de lenguaje.

Hay una serie de instrucciones llamadas instrucciones básicas que se utilizan de modo general en un algoritmo y que esencialmente soportan todos los lenguajes. Estas instrucciones básicas se clasifican en:

- Instrucciones de definición de datos.
- Instrucciones primitivas.
- Instrucciones compuestas.
- Instrucciones de control.

Aunque sólo vamos a utilizar el pseudocódigo para la representación de los algoritmos, vamos a mostrar también la representación de cada tipo de instrucción mediante el ordinograma.

4.3.- Sentencias de definición de datos.

Son aquellas instrucciones utilizadas para informar al procesador del espacio que debe reservar en memoria con la finalidad de almacenar un dato mediante el uso de variables simples o estructuras mas complejas.

La definición consiste en indicar un nombre a través del cual haremos referencia a un dato y un tipo a través del cual informaremos al procesador de las características y espacio que deberá reservar la memoria.

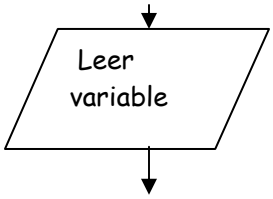
4.4.- Sentencias primitivas.

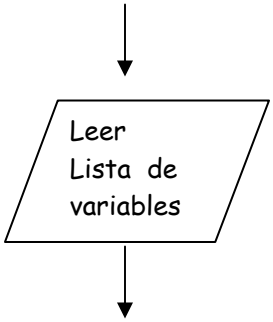
Son aquellas que ejecuta el ordenador de modo inmediato, ya que no dependen de otra cosa que de su propia aparición en el programa. Pueden ser : de entrada, de asignación y de salida.

4.4.1 Sentencias de entrada.

Son aquellas instrucciones encargadas de recoger datos de uno o varios dispositivos de entrada y almacenarlos en la memoria central en las variables que aparecen en la propia

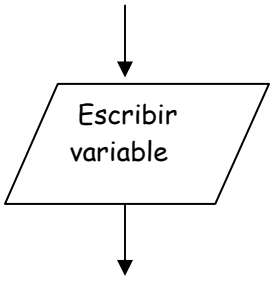
instrucción y que previamente han sido definidas. La representación mediante ordiograma y pseudocódigo es la siguiente:

Ordinograma	**Pseudocódigo**
	Leer variable

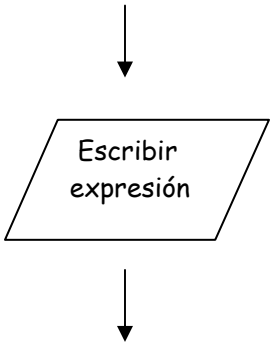
Ordinograma	**Pseudocódigo**
	Leer var1, var2, var3,..

4.4.2.- Sentencias de salida.

Son aquellas instrucciones encargadas de recoger los datos procedentes de variables (previamente definidas) a los resultados obtenidos de una expresión evaluada y depositarlos en un periférico o dispositivo de salida.

Ordinograma	**Pseudocódigo**
.....	
	Escribir variable
.....	.....

Ordinograma **Pseudocódigo**

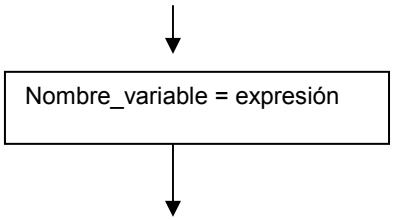


Escribir expresión

4.4.3.- Sentencias de asignación.

Son instrucciones que se utilizan para asignar valores a variables, o cambiar el valor almacenado en una variable obtenido de la evaluación de una expresión.

Ordinograma **Pseudocódigo**

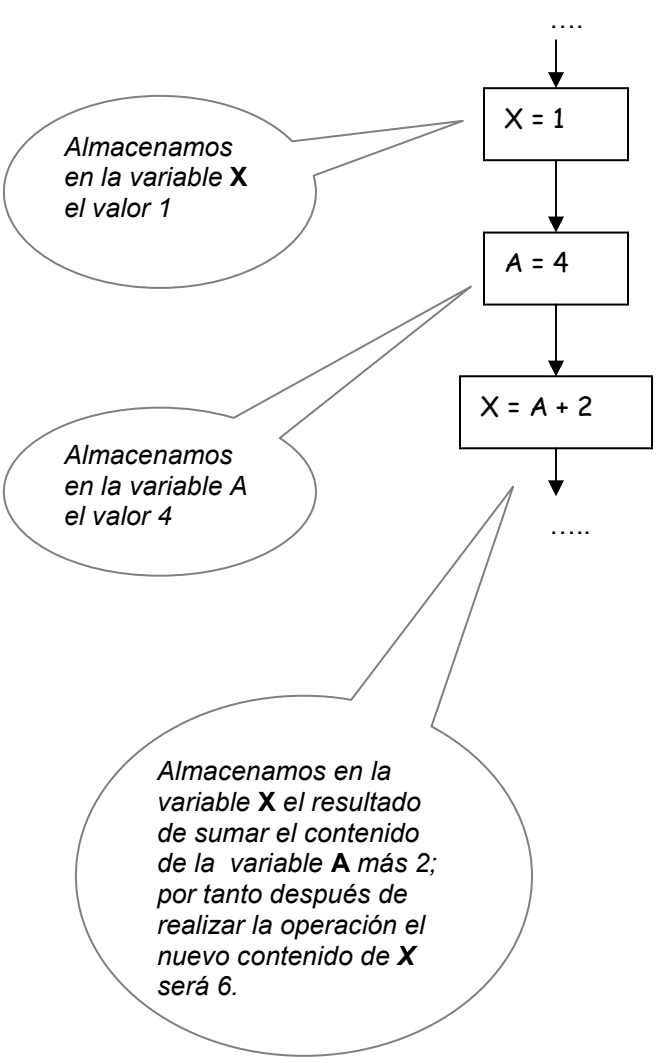


Nombre_variable ← expresión

La operación de asignación es destructiva ya que al almacenarse un nuevo valor en la variable se borra el que tenia antes.

Analicemos, por ejemplo, los contenidos de las variables después de realizar las siguientes instrucciones:

Ordinograma

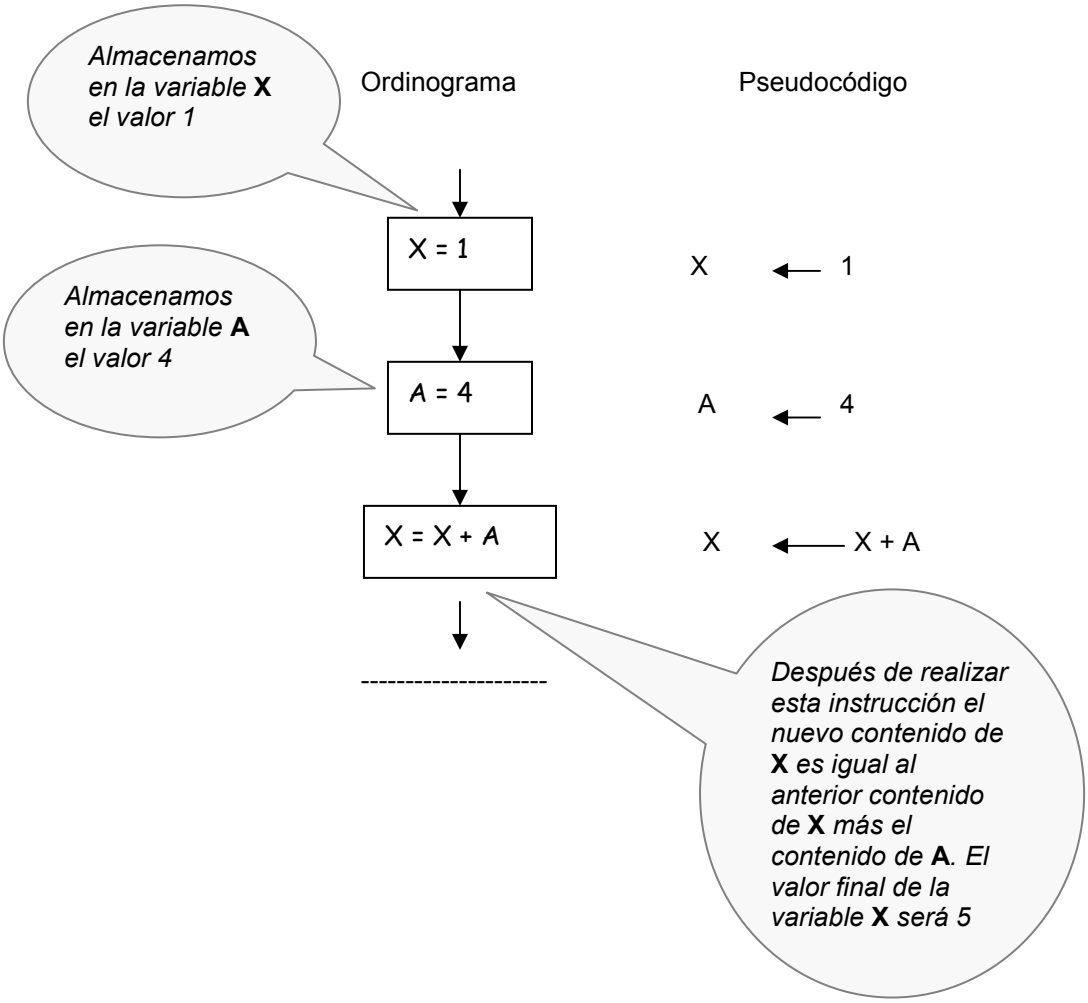


Pseudocódigo

```
........
X ← 1
A ← 4
X ← A + 2
.........
```

Es posible utilizar el mismo nombre de variable en ambos lados del operador de asignación.

Obsérvese, por ejemplo, este caso en el ejemplo siguiente:



Las sentencias de asignación se clasifican según el tipo de expresión en:

- Asignación aritmética: nota ← suma + 3
- Asignación lógica: numero ← 3 > 5
- Asignación alfanumérica: dia ← "hoy es martes"

En las operaciones de asignación hay que tener cuidado ya que no se puede asignar valores a una variable de diferente tipo.

Veamos algunos algoritmos de aplicación de estos tipos de instrucciones.

Práctica resuelta

Realizar el pseudocódigo de un programa que permita introducir por teclado el nombre y los apellidos de una persona y los muestre por pantalla. *Etiquetar* (poner mensajes aclaratorios que indiquen los datos que se están pidiendo o mostrando) tanto la entrada como la salida de resultados.

Analisis del problema

El algoritmo necesita que se le proporcionen al ordenador **tres variables de entrada** una para cada uno de los datos que pediremos por teclado. Esto lo realizamos mediante la instrucción **Leer** . Llamaremos a las variables Nombre (recogerá el nombre de la persona), apellido1 (para recoger el primer apellido) y apellido2 (para guardar el segundo apellido de la persona). Estos serán por tanto, los tres datos de entrada. El algoritmo tiene que **Visualizar** como datos de salida el contenido de la variable Nombre, el contenido de la variable apellido1 y el contenido de la variable apellido2.
También hemos etiquetado la salida de los resultados

La solución propuesta será la siguiente:

```
Programa resuelto datos

Entorno
      Nombre, apellido1, apellido2: cadena de caracteres
Algoritmo


Inicio
                                  Visualizar “Introduce el nombre”
                                  Leer Nombre

                                  Visualizar “Introduce el primer apellido”
                                  Leer apellido1

                                  Visualizar “Introduce el segundo apellido”
                                  Leer apellido2

                                  Visualizar “El nombre es”, Nombre
                                  Visualizar “El primer apellido es”, apellido1
                                  Visualizar “El segundo apellido es”, apellido2

Finprograma
```

Sentencias de definición de datos, en las que asignamos un nombre a cada una de las variables (Nombre, apellido1, apellido2) a través del cual haremos referencia a un dato y un tipo (cadena de caracteres) e informaremos al procesador de las características y espacio que deberá reservar en la memoria.

Entre comillas escribimos el texto aclaratorio que queremos que se muestre en pantalla. Éste indica que estamos pidiendo el nombre de la persona. Es lo que antes hemos definido como *etiquetar.*

Los nombres de las variables en las que recogemos los datos que pedimos como entrada, se esciben tal y como los definimos anteriormente en el Entorno

Visualizamos el contenido de las variables. Escribimos entre comillas el texto que queremos que aparezca en pantalla y fuera de las comillas el nombre de la variable que guarda el dato, así nos mostrará su contenido.

Prácticas propuestas

Práctica 4-1.

Realizar el pseudocódigo de un programa que permita calcular la suma de los cuadrados de dos números enteros que introduciremos por teclado. Etiquetar tanto la entrada de datos como la salida de resultados.

Práctica 4-2.

Realizar el pseudocódigo de un programa que permita calcular el 10% de una determinada cantidad entera que introduciremos por teclado. Etiquetar tanto la entrada de datos como la salida de resultados.

Práctica 4-3.

Realizar el pseudocódigo de un programa que permita emitir facturas a partir del precio (numércico real) de un articulo (sin IVA) teniendo en cuenta que se hace un descuento del 15% sobre todos los artículos. El IVA se carga sobre el precio inicial.

Precio.....: xxxx €
IVA .....: xxxx €
Descuento(15%) .....: xxxx €
Precio final .....: xxxx €

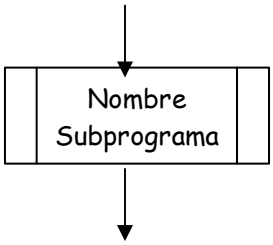
Práctia 4-4.

Realizar el pseudocódigo de un programa que permita leer el nombre de un alumno y las notas (numéricas enteras) de las tres evaluaciones mostrando al final la nota media del curso. Etiquetar tanto la entrada de datos como la salida de resultados.

4.5.- Sentencias compuestas.

Son aquellas instrucciones que no pueden ser ejecutadas directamente por el procesador, y están constituidas por un bloque de acciones agrupadas en subrutinas, subprogramas, funciones o módulos.

Ordinograma



pseudocódigo

```
Programa nombre_programa
  ----
  nombre_subprograma
  -------
fin-programa

nombre_subprograma
  instrucciones
fin-subprograma
```

4.6.- Contadores, acumuladores, interruptores.

Los contadores, acumuladores e interruptores son elementos auxiliares de un programa.

4.6.1.- Contadores.

Un contador es una variable que se utiliza para contar, es decir que su valor se incrementa o decrementa en una cantidad fija cada vez que se ejecuta la instrucción que lo contiene.

Identificador ← Identificador ± constante

Se utilizan en las estructuras repetitivas para contar el nº de veces que queremos se ejecute un conjunto de instrucciones o para contar un suceso particular solicitado por el enunciado del problema.

Contador ← contador + 1

Los contadores siempre hay que inicializarlos es decir darles un valor inicial (generalmente 0) antes de ser utilizado. Contador ← V_i

6.4.2.- Acumuladores.

Son variables que se utilizan para almacenar cantidades variables resultantes de operaciones sucesivas.

Identificador1 ← identificador1 + identificador2

Identificador1 ← identificador1 * identificador2

Los acumuladores también hay que inicializarlos y será con el valor neutro de la operación que se va a realizar: 0 en el caso de la suma, 1 en el caso del producto.

4.6.3.- Interruptores.

Los interruptores, también llamados conmutadores, switches o indicadores son variables que solo pueden tomar dos valores considerados como lógicos y opuestos entre sí a lo largo de la ejecución del programa. Sw = 0 , 1 ; Sw = F , V ; Sw = N , S

Realizan la función de transmitir información de un punto a otro del programa. Hay que darles un valor inicial e irán cambiando de valor en los puntos adecuados del programa, de forma que examinando su valor podemos realizar la transmisión de información que queríamos. También se utilizan para realizar procesos de forma alternativa.

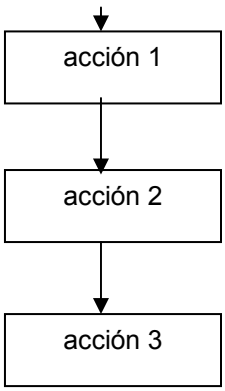
4.7.- Sentencias de control.

Son instrucciones que controlan el flujo de ejecución de otras instrucciones. También se llaman instrucciones estructuradas por que utilizan una serie de estructuras que son las siguientes:

4.7.1.- Sentencias secuenciales.

Es un conjunto de instrucciones que se ejecutan unas a continuación de otras, en el orden en que están escritas en el programa.

Ordinigrama **Pseudocódigo**



acción 1

acción 2

.

acción n

Práctica resuelta

Realizar el pseudocódigo de un programa que permita realizar la suma de dos números que introduciremos por teclado. Etiquetar tanto la entrada de datos como la salida de resultados.

Analisis del problema

Para solucionar el algoritmo se necesita proporcionar al ordenador dos variables de entrada a las que llamaremos num1 para guardar el dato del primer número y num2 para guardar el segundo número (ambos los introduciremos por teclado) Como dato de salida definimos la variable suma, que será donde guardemos el resultado de la operación a realizar. El programa deberá realizar la suma aritmética de ambos números. La operación la realizamos mediante la instrucción **suma ← num1+ num2;** ésta la interpretaremos del siguiente modo: acumulamos el dato que está en la variable num1, al contenido de la variable num2, y el resultado lo guardamos en la variable suma. Por último visualizamos el contenido de la variable suma, que es la que guarda el resultado. El conjunto de instrucciones se ejecutarán secuencialmente, es decir una tras otra.

La solución propuesta será la siguiente:

```
Programa resuelto calculos
Entorno
        num1, num2, suma : numéricos enteros
Algoritmo
Inicio
        Visualizar "Introduce el primer número"
        Leer num1

        Visualizar "Introduce el segundo número"
        Leer num2

        suma ← num1 + num2

        Visualizar "La suma es", suma
Finprograma
```

Etiquetamos la entrada de datos y para ello escribimos entre comillas el texto que queremos que se muestre en pantalla. Éste indica que estamos pidiendo el primer número.

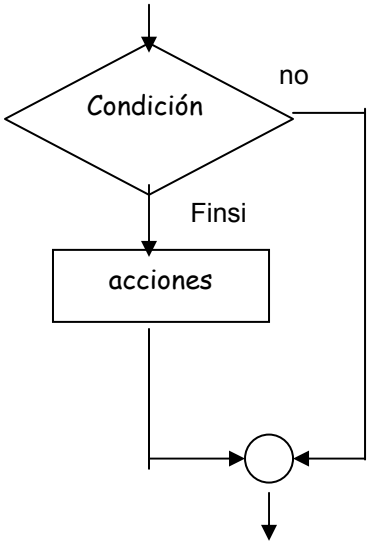
Visualizamos el contenido de la variable suma. Escribimos entre comillas el texto que queremos que aparezca en pantalla y fuera de las comillas el nombre de la variable que guarda el resultado, así nos mostrará su contenido.

4.7.2.- Sentencias alternativas.

Controlan la ejecución o la no ejecución de uno o varios bloques de instrucciones dependiendo del cumplimiento o no de una condición o del valor de una expresión. Pueden ser:

- **Simples:** controlan la ejecución de un conjunto de instrucciones por el cumplimiento o no de una condición. Si la condición se cumple se ejecuta el grupo de instrucciones y si no se cumple no se ejecuta nada.

Ordinograma

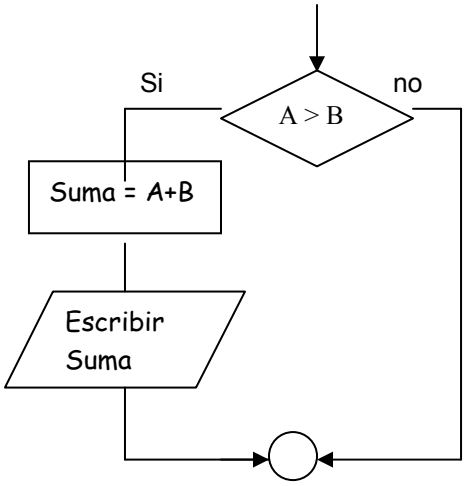


Pseudocódigo

```
si <condición> entonces
        <acciones>
fin-sí
```

Ejemplo

Ordinograma

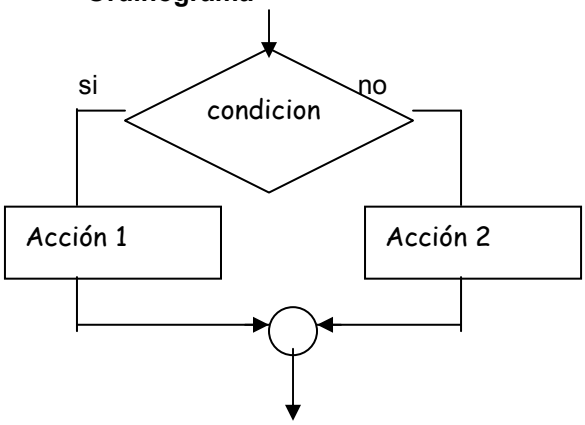


Pseudocódigo

```
si A > B entonces
        Suma ← A + B
        escribir "La suma es: ", Suma
finsi
```

- **Dobles:** controla la ejecución de dos conjuntos de instrucciones dependiendo del cumplimiento o no de una condición. Si se cumple se ejecutan unas instrucciones y si no se cumplen se ejecutan otras.

Ordinograma

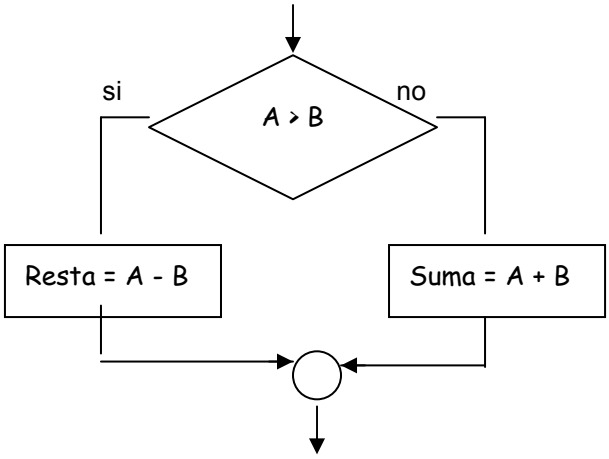


Pseudocódigo

```
si <condicion> entonces
        <accion 1>
sino
        <accion 2>
finsi
```

Ejemplo

Ordinograma

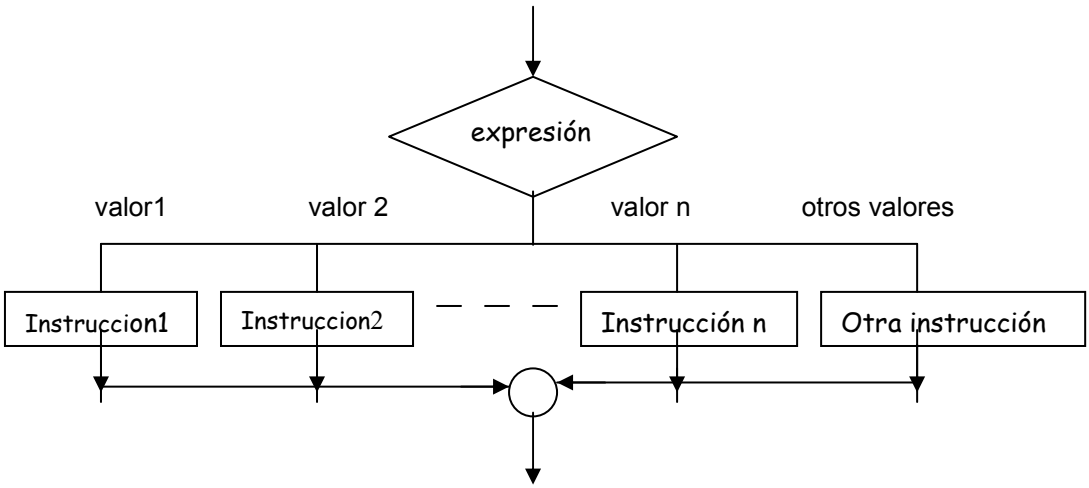


Pseudocódigo

```
si A > B  entonces
        Resta ← A - B
sino
        Suma ← A + B
finsi
```

- **Múltiples:** controla la ejecución de varios conjuntos de instrucciones, por el valor de una expresión, de forma que cada conjunto de instrucciones está ligado a un posible valor de dicha expresión.
  Se ejecutará el conjunto de instrucciones que se encuentra relacionado con el valor que resulte de evaluar la expresión.
  Las distintas opciones tienen que ser disjuntas, es decir, solo pueden ejecutarse un conjunto de instrucciones a la vez.

Ordinograma



Pseudocódigo

```
Segun_valor expresión

        valor 1: instrucción 1
        valor 2: instrucción 2
         .....
        valor n: instrucción n
        sino: otra instrucción

fin segun_valor
```

Práctica resuelta

Hacer el pseudocódigo de un programa que lee un número por teclado y comprueba e imprime en los dispositivos estándar de salida si dicho número es nulo.

Análisis del problema

Se debe leer un número que guardaremos en la variable Num. Éste será el dato de entrada y que tendremos que definir en el Entorno. Como dato de salida se visualizará un mensaje que nos indicará si el número es nulo o no. Se trata de comparar el contenido de la variable Num con cero y mostrar el correspondiente mensaje dependiendo de si la respuesta ha sido SI o NO.

La solución propuesta sería la siguiente

Pseudocódigo

```
Programa Numeros nulos

Entorno
        Num: numérico entero

Algoritmo

Inicio
        Visualizar "Introduce un número"
        Leer Num
        Si      Num = 0 entonces
                Visualizar "El número ", Num, "es nulo"
        Sino
                Visualizar "El número ", Num, "no es nulo"
        Finsi
Finprograma
```

Sentencia de definición de datos, en la que asignamos un nombre a la variable que va a guardar el número que introduciremos por teclado. En este caso le hemos asignado el nombre Num y con él hacemos referencia a un dato y un tipo (numérico entero) e informaremos al procesador de las características y espacio que deberá reservar en la memoria.

Entre comillas escribimos el texto aclaratorio que queremos que se muestre en pantalla. Éste indica que estamos pidiendo un número. Recuérdese que esto lo hemos definido como *etiquetar* los datos.

El nombre de la variable en la que recogemos el datos que pedimos como entrada, se escribe tal y como lo definimos anteriormente en el Entorno

Evaluamos el contenido de la variable Num. **Si** el contenido de Num es 0 **entonces** visualizamos el mensaje que nos lo advierte. **Sino** visualizamos el otro mensaje. Nótese que la estructura finaliza con **finsi**

Visualizamos el contenido de la variable. Escribimos entre comillas el texto que queremos que aparezca en pantalla y fuera de las comillas el nombre de la variable que guarda el dato, así nos mostrará su contenido. En este caso el mensaje nos avisa de si el contenido de la variable Num es nulo o no.

Práctica resuelta

Realizar el pseudocódigo de un programa que tenga como pantalla de entrada:

Quieres continuar (s/n).....: _

Y como pantalla de salida:

La respuesta es correcta y es: _

O

La respuesta es incorrecta no es ni un "s" ni una "n".

Teniendo en cuenta las mayúsculas y las minúsculas.

Análisis del problema

El algoritmo debe comenzar visualizando un mensaje por pantalla que nos pregunte "Quieres continuar (s/n)?". Como dato de entrada declaramos una variable de tipo carácter porque será la que guarde la respuesta que se leerá por teclado. Llamaremos a esta variable **respuesta** Después el algoritmo debe comprobar que la entrada es correcta. Sólo se considerarán correctas la "**s**", la "**S**", la "**n**" y la "**N**", ya que se tendrán en cuenta las mayúsculas y las minúsculas. Como se trata de caracteres alfabéticos la variable **respuesta** tenemos que definirla de tipo carácter.

La solución propuesta será la siguiente

Sentencia de definición de datos, en la que asignamos un nombre a la variable que va a guardar la respuesta por teclado. En este caso le hemos asignado el nombre respuesta y el tipo carácter.

El operador lógico O, nos permite incluir en la sentencia varias condiciones y en este algoritmo significa que *con que se cumpla una de las condiciones la respuesta será válida*

Evaluamos el contenido de la variable respuesta. **Si** el contenido de respuesta es "s" o "S" o "n" o "N" **entonces** visualizamos el mensaje que nos informa de que hemos respondido correctamente. **Sino** visualizamos el mensaje contrario.

```
Programa ejercicio Continuar
    Entorno
        respuesta: carácter
    Algoritmo
    Inicio
        Visualizar "Quieres continuar (s/n) "
        Leer respuesta
        Si (respuesta = 's' ) o (respuesta = 'S' ) o  (respuesta  = 'n' )  o
        (respuesta = 'N' ) entonces
            Visualizar "La respuesta es correcta y es: ",  respuesta
        sino
            Visualizar " La respuesta es incorrecta, no es ni s/S ni n/N"
        finsi
    Finprograma
```

Práctica resuelta

Realizar el pseudocódigo de un programa que a partir del número de día de la semana, nos visualice en pantalla el correspondiente día de la semana alfabéticamente. Utilizar la Sentencia de Alternativa Múltiple *Según valor.*

Análisis del problema

Como dato de entrada sólo tendremos un número comprendido entre 1 y 7. Como dato de salida, la conversión de ese número a un día de la semana. Para resolver el problema supondremos que el número introducido siempre estará entre el rango de valores. Así, definimos una variable numérica entera que almacenará el número que introducimos por teclado. Llamaremos a esta variable **dia**. A continuación evaluaremos el valor de la variable **dia**, y según sea su valor escribiremos en pantalla Lunes, Martes, Miércoles,…

La solución propuesta será la siguiente

```
Programa Dia de la Semana
Entorno
        dia: numerico entero
Algoritmo
Inicio
        Visualizar “Introduce un numero de dia”
        Leer dia
        Según_valor dia hacer
                1: Visualizar “Lunes”
                2: Visualizar “Martes”
                3: Visualizar “Miércoles”
                4: Visualizar “Jueves”
                5: Visualizar “Viernes”
                6: Visualizar “Sabado”
                7: Visualizar “Domingo”
        Fin según_valor
Finprograma
```

Sentencia de definición de datos, en la que asignamos un nombre a la variable que va a almacenar el valor (comprendido entre 1 y 7) que introduciremos por teclado. En este caso le hemos asignado el nombre **dia** y el tipo numérico entero.

Se ejecutará el conjunto de instrucciones que se encuentra relacionado con el valor que resulte de evaluar la expresión. Según sea el valor de la variable **dia** visualizará un mensaje u otro. Si el valor de la variable **dia** es 1, Visualizará Lunes; Si el valor de la variable **dia** es 2 visualizará Martes…. y así hasta cumplimentar la semana.

Práctica resuelta

Realizar un programa que determine el precio de un billete de ida y vuelta en ferrocarril, conociendo la distancia a recorrer y sabiendo que si el número de días entre la ida y la vuelta es superior a 7 y la distancia es superior a 800 Km., el billete tiene una reducción del 30%. El precio del Km. es de 0.15 céntimos de €. Los datos se introducirán por teclado y los resultados se mostrarán en pantalla.

Análisis del problema

Los datos de Entrada del problema son la distancia a recorrer y el número de días de estancia. Asignaremos a las variables los nombres de **distancia** y **dias**, respectivamente. Como dato de Salida definimos el **precio** del billete. Se lee la distancia y el número de días. Ambos valores se guardan en las variables que hemos definido para tal fin: **distancia** y **dias**
Se halla el precio del billete de ida y vuelta **precio** ← **distancia** * 2 * 0.15. Después se comprueba si la distancia es superior a 80 Km. y los días de estancia a 7 y si es cierto se aplica una reducción del 30%.

```
Programa Cálculo del precio
Entorno
        precio, distancia: numéricos reales
        dias: numerico entero
Algoritmo
Inicio
        Visualizar "Introduce la distancia en  Kms"
        Leer distancia
        Visualizar "Introduce numero de dias"
        Leer dias
        precio ← distancia * 2 * 0.15
        si (dias > 7) y (distancia > 800) entonces
                precio ← precio * 0.3
        finsi
        Visualizar "El precio del viaje es de ", precio " €uros"
Finprograma
```

Mediante esta sentencia alternativa se comprueba el contenido de las variables **dias** y **distancia**. Nótese que se ha utilizado el operador lógico **y** entre ambas condiciones , porque es necesario que se cumpla una condición **y** la otra para realizar el descuento del 30%.

Prácticas propuestas a resolver por el alumno.

Práctica 4-5

Hacer el pseudocódigo de un programa que pide un número por teclado y comprueba si es nulo, positivo o negativo, visualizando el resultado en pantalla, con los mensajes oportunos.

Práctica 4-6

Realizar un programa que lea tres números distintos por teclado y nos diga cual de ellos es el mayor.

Práctica 4-7

Realizar un programa que recibe como dato de entrada una hora expresada en horas, minutos y segundos y calcule y escriba la hora, minutos y segundos que serán transcurrido 1 segundo.

Práctica 4-8

Realizar un programa que lea un número por teclado y compruebe si está entre 10 y 100 ambos inclusive. Visualizar en pantalla los mensajes oportunos.

Práctica 4-9

Hacer el pseudocódigo de un programa que lee por teclado la nota entera de un alumno comprendida entre 0 y 10 y la transforma en su equivalente alfabética. Mostrar el resultado por pantalla. (0, 1, 2, y 4 → Insuficiente; 5→ Suficiente; 6→ Bien; 7,8→Notable; 9, 10→Sobresaliente). Visualizar en pantalla los mensajes correspondientes.
Visualizar en pantalla los mensajes correspondientes.

Prácticas propuestas a resolver por el alumno.

Práctica 4-10

Realizar un programa que permita introducir un número entero por teclado y nos muestre en pantalla si es par o impar.

Nota: La mayoría de los lenguajes de programación disponen de una función MOD, que nos permite quedarnos con el resto de la división de dos números enteros. De este modo si hacemos C= A MOD B, en la variable C tendríamos el resto de hacer la división entera de A entre B

Práctica 4-11

Realizar un programa que permita introducir un número entero entre 1 y 10 y nos visualice en pantalla si dicho numero es par o impar. En el caso de que el nº no esté dentro del rango, se visualizará un mensaje de error y se terminará el programa.

.

4.7.3.- Sentencias de iteración o repetitivas.

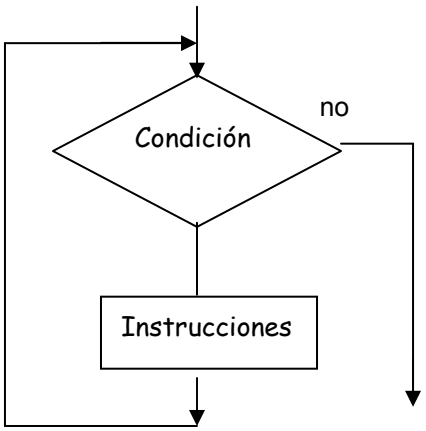
Controlan la ejecución repetitiva de un conjunto de instrucciones mediante una condición que determina el número de veces que se ha de repetir esta ejecución.

El valor de la condición tiene que estar afectado por la ejecución de las instrucciones para asegurar la terminación de esta repetición.

Existen varios tipos de estructuras repetitivas:

- Estructura **mientras**: controla la ejecución de un conjunto de instrucciones de tal forma que estas se ejecutan mientras se cumpla la condición, que será evaluada siempre antes de cada repetición.
  Puede que las instrucciones no se ejecuten nunca.

Ordinograma



Pseudocódigo

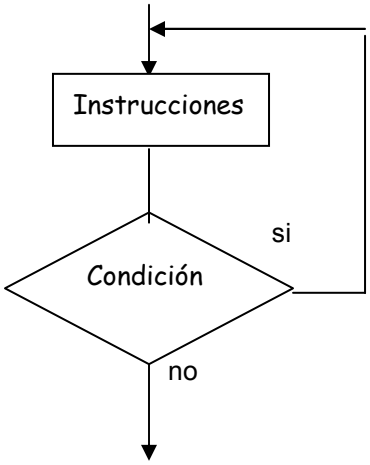
```
Mientras condición

        Instrucción 1
        Instrucción 2
        ……..
        Instrucción n

finmientras
```

- Estructura **repetir** : controla la ejecución de un conjunto de instrucciones, de tal forma que estas se ejecutan mientras que se cumpla la condición que será evaluada siempre después de cada repetición.
  Las instrucciones se ejecutan al menos una vez.

Ordinograma **Pseudocódigo**



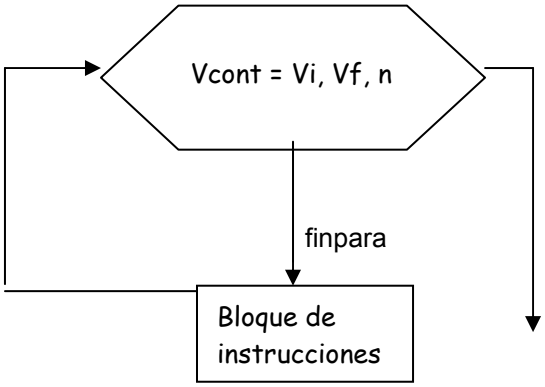

```
repetir
        instrucción 1
        instrucción 2
        .....
        instrucción n
mientras condición
```

- Estructura **para:** controla la ejecución de un conjunto de instrucciones, de tal forma que estas se ejecutan un número determinado de veces que se conoce de antemano.

  Esta estructura lleva asociada una variable que actúa como contador. El contador parte de un valor inicial y se irá incrementando en cada repetición hasta llegar al valor final que es cuando se detiene la ejecución.

  A este contador se le llama variable de control.

Ordinograma



Vcont = variable de control

Vi = valor inicial

Vf = valor final

Inc = incremento

Nº de repeticiones = (Vf – Vi) div Inc + 1

Pseudocódigo

```
Para Vcont de Vi a Vf con Inc=n

        instrucción 1
        instrucción 2
        .....
        instrucción n

finpara
```

4.7.4.- Sentencias de salto.

Son aquellas instrucciones que alteran o rompen la secuencia normal de ejecución de un programa perdiendo toda posibilidad de retornar el control de ejecución del programa al punto de llamada. El uso de este tipo de instrucciones debe quedar restringido en una programación estructurada.

Pseudocódigo:

```
ir_a etiqueta.
```

Las instrucciones de salto se pueden clasificar de la manera siguiente:

- Instrucciones de salto condicional: Son aquellas que alteran la secuencia de ejecución de un programa sólo y exclusivamente en el caso de que la condición especificada sea cierta.
- Instrucciones de salto incondicional: Alteran la secuencia normal de ejecución de un programa siempre, pues no van acompañadas de una condición que limite en determinadas ocasiones la realización del salto a otra parte del programa.

Práctica resuelta

Diseñar un algoritmo para calcular la suma de los 10 números naturales (del 1 al 10). Visualizar el resultado en pantalla.

Análisis del problema utilizando la **"estructura para"**

Resolveremos el problema analizando cada uno de los tres métodos posibles para observar las diferencias entre unas y otras.

En este problema no hay una entrada de datos externa, porque los números que vamos a sumar los genera la propia **estructura para**. Para realizar la suma de los 10 números será imprescindible realizar un bucle que se ejecute 10 veces (tantos como números a sumar) y que incluya dentro de él un acumulador (en nuestro caso hemos llamado a la variable acumulador **suma**). Como conocemos de antemano el número de veces que se va a ejecutar el bucle (10), es muy adecuado utilizar un bucle de este tipo.

La solución propuesta utilizando la **"estructura para"** será la siguiente:

```
Programa Estructura Para
Entorno
        I, suma : numericos enteros
Algoritmo
Inicio
        suma ← 0
        Para I desde 1 hasta 10 con incremento 1 hacer
                suma ← suma + I
        Finpara
        Visualizar "La suma es ", suma
Finprograma
.
```

Inicializamos el acumulador a 0, para evitar el riesgo de que esa variable contenga información. Aunque en este caso no es necesario, es una práctica muy aconsejable la de **inicialización de las variables**.

Esta estructura lleva asociada una variable de control a la que hemos llamado **I**, que es la que actúa como contador. Este contador parte de un valor inicial, en este caso 1 y se irá incrementando de 1 en 1 en cada repetición hasta llegar al valor final. La instrucción suma ← suma + I está dentro del bucle, y por tanto se ejecutará tantas veces como indique la condición. En este caso 10 veces.

Cuando el contador llegue a 11, **ya no entrará en el bucle** y ejecutará la siguiente instrucción para visualizar el total de la suma, que está alamacemado en la variable **suma**.

Análisis del problema utilizando la **"estructura mientras"**

En este algoritmo no hay una entrada de datos externa, porque los números que vamos a sumar los genera el contador identificado con el nombre **I**. Dentro del bucle Mientras utilizado ejecutaremos dos instrucciones:

La 1ª **suma ← suma + I** lo que hace es acumular en la variable suma el resultado de sumar 1+2+3+…+10.

La 2ª **I ← I + 1** es la que hace que la variable I se vaya incrementando de 1 en 1.

La propia **estructura mientras** se encarga de evaluar la condición que en este caso establecemos como **I < = 10.**

La solución propuesta utilizando la **"estructura mientras"** será la siguiente:

```
Programa Estructura Mientras
Entorno
        I, suma : numericos enteros
Algoritmo
Inicio
        suma ← 0
        I ← 1
        Mientras I <= 10 hacer
                suma ← suma + I
                I ← I + 1
        Finmientras
        Visualizar "El valor de la suma es: ", suma
Finprograma
```

La variable **I**, actuará como contador del número de números que se van a acumular en la variable **suma.** En esta instrucción se inicializa en 1.

En el bucle **Mientras**, **I** controla la ejecución de las dos instrucciones de tal forma que estas se ejecutan mientras se cumpla la condición, que será evaluada siempre antes de cada repetición. Mientras **I** sea menor o igual que 10, se seguirán ejecutando las instrucciones

suma ← suma + I

I ← I + 1

Cuando **I** tome el valor 11, el programa se saldrá del bucle para visualizar el contenido de la variable **suma**

Análisis del problema utilizando la **"estructura repetir"**

En este caso, las instrucciones que están dentro de la estructura repetitiva, se ejecutan al menos una vez, pues la condición para salir del bucle se evalúa al final.

Las instrucciones que están dentro del bucle, son las mismas que en la estructura "mientras".

La solución propuesta utilizando la **"estructura repetir"** será la siguiente:

```
Programa Estructura Repetir
Entorno

        I, suma : numéricos enteros
Algoritmo
Inicio
        suma ← 0
        I ← 1
        Repetir
                suma ← suma + I
                I ← I + 1
        Mientras I < 11
        Visualizar "el valor de la suma es: ", suma
Finprograma
```

La variable **I**, actuará como contador del número de números que se van a acumular en la variable **suma.** En esta instrucción se inicializa en 1.

Dentro de la estructura se encuentran las dos intrucciones que la componen. La que acumula la suma en la variable **suma.**, y la que se encarga de ir incrementando de 1 en 1 el contador I, que será el que regule el nº de números a sumar.
Una vez realizadas estas dos instrucciones es cuando se evalúa la condición para continuar dentro del bucle. Cuando **I** tome el valor 11, el algoritmo se sale de la estructura repetitiva para visualizar el contendio de la variable **suma**.

Prácticas propuestas a resolver por el alumno.

Práctica 4-12

Escribir el algoritmo de un programa que lea e imprima una serie de números distintos de cero. El programa debe terminar con un valor cero que no se debe imprimir. Finalmente se desea obtener la cantidad de valores leídos distintos de cero. Utilizar la "estructura mientras". Etiquetar los datos de entrada y de salida.

Práctica 4-13

Escribir el algoritmo de un programa que imprima y sume la serie de números 3, 6, 9, 12, ..., 99. Etiquetar los datos de entrada y de salida. Utilizar la sentencia repetitiva ***repetir.***

Práctica 4-14

Escribir el algoritmo de un programa para determinar si un número n es primo (un número primo sólo es divisible por el mismo y por la unidad). Etiquetar los datos de entrada y de salida. Utilizar la sentencia repetitiva ***mientras.***

Práctica 4-15

Hacer el algoritmo de un programa que lea un número entero positivo N y calcule e imprima su factorial N! Teniendo en cuenta que el factorial se calcula así:

0! = 1
1! = 1
2! = 2 * 1
3! = 3 * 2 * 1
4! = 4 * 3 * 2 * 1
.
N! = N * (N-1) * (N-2) * (N-3) * ... * 3 * 2 * 1

Prácticas propuestas a resolver por el alumno.

Práctica 4-16

Realizar el algoritmo de un programa que nos pida un nº entero entre 1 y 9 y que escriba la tabla de multiplicar. Etiquetar la información de entrada y de salida.

Práctica 4-17

Hacer el algoritmo de un programa que lea 10 números enteros y escriba si hay algún 5. Etiquetar la información de entrada y de salida.

Objetivos

Al finalizar esta unidad el alumno será capaz de resolver problemas desde el punto de vista informático con un algoritmo y será capaz también de proponer soluciones en forma de algoritmos. El tema está diseñado además para que, mediante la práctica, consiga aprender a diseñar algoritmos por si mismo.

- Entender, analizar y resolver una amplia variedad de problemas computacionales.
- Diseñar e implementar soluciones eficientes y de calidad a los problemas como resultado de la aplicación de un proceso metódico.
- Conocer el concepto de Algoritmo y su importancia en el mundo de las aplicaciones para ordenadores.
- Manejar con soltura la lógica de programación.
- Distinguir los diferentes tipos de algoritmos que existen y sus variantes.
- Conocer los conceptos sobre las técnicas de diseño.
- Saber diseñar algoritmos eficientes mediante la programación estructurada.
- Conocer y saber aplicar las reglas de programación estructurada.
- Saber hacer un uso eficiente de las estructuras de control de flujo.
- Realizar tareas de investigación y búsqueda de información adicional en Internet.

Enlaces de interés

☞ Sitio muy interesante sobre programación estructurada. En esta página se hace un breve comentario sobre los tipos de estructuras de control de flujo que podemos encontrar.

Definición de las Estructuras Básicas de Control.
http://www.itlp.edu.mx/publica/tutoriales/pascal/u1_1_4.html

☞ Algunos algoritmos resueltos que pueden ayudarte a entender la resolución de problemas y el diseño de algoritmos. En el siguiente enlace vas a encontrar un resumen del tema de algoritmos, con ejemplos de elaboración y al final unos cuantos ejemplos resueltos.

Teoría de Algoritmos.
http://www.geocities.com/inf135/tutc/Tema02.htm

☞ Si te interesa el tema puedes ampliar conocimientos en el siguiente enlace, en el que encontrarás algunas sugerencias y reglas que te ayudarán como futuro programador.

Curiosidades sobre programación
http://es.wikipedia.org/wiki/Paradigma_de_programación

Glosario de términos

Aplicación

En informática, programa o conjunto de programas cuya función es automatizar una o varias tareas.

Características Físicas de los Soportes.

Cómo se representa realmente la información (datos) sobre la memoria de la máquina.

Datos Globales

En programación. Datos disponibles desde cualquier módulo del programa, en nuestro caso, serían los disponibles desde cualquier parte del módulo del programa.

Dependencias Funcionales

Las dependencias funcionales son una restricción al conjunto de relaciones legales. Relación entre los resultados de los datos y los datos.

Encapsulación.

Programación. También llamada "ocultación de la información", asegura que los objetos no pueden cambiar el estado interno de otros objetos de maneras inesperadas; solamente los propios métodos internos del objeto pueden acceder a su estado. Cada tipo de objeto expone una *interfaz* a otros objetos que especifica cómo otros objetos pueden interactuar con él.

Estructura de Datos

Forma de organizar y representar los datos para un problema. En programación, está muy relacionada con la forma de representar internamente, en memoria, los datos en el ordenador.

Función.

Igual que el procedimiento pero con la diferencia que devuelve un valor. Ver procedimiento

Interfaz o Interface

Informática: Zona de comunicación o acción de un sistema sobre otro
En programación orientada a objetos: Un interface es una colección de definiciones de métodos (sin implementaciones) y de valores constantes.

Instancia

Cada una de las preguntas relacionadas en las que se puede dividir un problema. En programación orientada a objetos, característica o parte de una clase.

Paquete.

Informática. Programación. Un conjunto de estructuras de datos y subprogramas que tienen cierta relación y coherencia entre sí.

Procedimiento.

Subprograma o segmento de código que ejecuta una determinada tarea y luego devuelven el procesamiento al área de código desde donde fue llamado.

Subrutina.

Una subrutina o subprograma, como idea general, se presenta como un algoritmo separado del algoritmo principal, el cual permite resolver una tarea específica.

Tipos de Datos Abstractos

Modelos matemáticos sobre los que se definen una serie de operaciones. El tipo abstracto de datos es una colección de valores y operaciones que se definen mediante una especificación que es independiente de cualquier representación.

Código reutilizable

Código que podemos utilizar en otros programas, funciones, etc...

Contador

Variable que acumula el número de ocurrencias de una determinada acción.

Eficiencia de los algoritmos

Se dice que un algoritmo es eficiente cuando consume pocos recursos durante su ejecución, estos recursos son tiempo del procesador y espacio en memoria.

Entorno volátil de un programa

Está constituido por las partes del procesador que no pueden ser compartidas (el contador de programa, registros acumuladores, registro de estado, ...) Es la información que se guarda cuando un proceso pierde el control de la CPU para poder continuar con su ejecución correctamente más adelante.

Excepción

Una excepción es un evento que ocurre durante la ejecución del programa que interrumpe el flujo normal de las sentencias.

Función

Una función es una REGLA o CRITERIO para obtener un cierto resultado a partir de unos valores determinados.

Función EOF()

La función eof (*end of file*), fin de archivo, devuelve el estado de un archivo. Es una función de tipo lógico que indica si el fin de archivo se ha encontrado; devuelve true si se encontró, false en caso contrario.

Indentación

Tabulación del código de un programa para hacerlo más legible.

Lenguaje Basic

Lenguaje de programación de alto nivel. El nombre de BASIC, significa *Beginner's All-purpose Symbolic Instruction Code* (Código de Instrucción Simbólico de Propósito General para Principiantes)

Método

Nombre dado a los subprogramas que permiten el manejo de los objetos bajo el paradigma de la Programación Orientada a Objetos.

Paradigmas de programación

Un paradigma está constituido por los supuestos teóricos generales, las leyes y las técnicas para su aplicación que adoptan los miembros de una determinada comunidad científica, en nuestro caso concreto representan un enfoque particular o filosofía para la construcción del software.

Procedimiento

Un procedimiento es un subprograma que realiza una tarea específica Se llama así a un subprograma que ejecuta unas ciertas acciones sin que valor alguno de retorno esté asociado a su nombre. En otras palabras: NO devuelven valores (en cierto sentido).

Programa autodocumentado

Programa que utiliza la indentación, los comentarios, tiene especial cuidado en la utilización con claridad de los comandos y la significación de los nombres de las variables.

Programación modular

Técnica de programación en la que un programa se divide en módulos, que resultan de segmentar el problema en funciones lógicas que son perfectamente diferenciables. Cada uno de los módulos ejecuta una única actividad o tarea, y se analiza, se codifica, y "optimiza" independientemente de otros módulos.

Programación orientada a objetos:

Programación que utiliza lenguaje basado en objetos. Se define un objeto como un conjunto complejo de datos y programas que poseen estructura y forman parte de una organización determinada.

Sentencias o estructuras de salto incondicional

Una sentencia de salto incondicional (*goto*) continúa la ejecución de un programa en un punto especificado por una etiqueta diferente al camino natural del código.

Subrutina

En computación, una subrutina o subprograma, como idea general, se presenta como un algoritmo separado del algoritmo principal, el cual permite resolver una tarea específica.

Teorema de programación estructurada

Se ha demostrado que un programa propio puede ser escrito utilizando solamente tres tipos de estructuras de control -secuenciales, selectivas y repetitivas-. Un programa es propio si posee un sólo punto de entrada y uno de salida, si existen caminos desde el inicio hasta el fin que se pueden seguir y que pasan por todas las partes del programa, y si todas las instrucciones son ejecutables sin que hayan bucles infinitos.

Valor resultado de un subprograma

Valor resultante de la ejecución de un subprograma