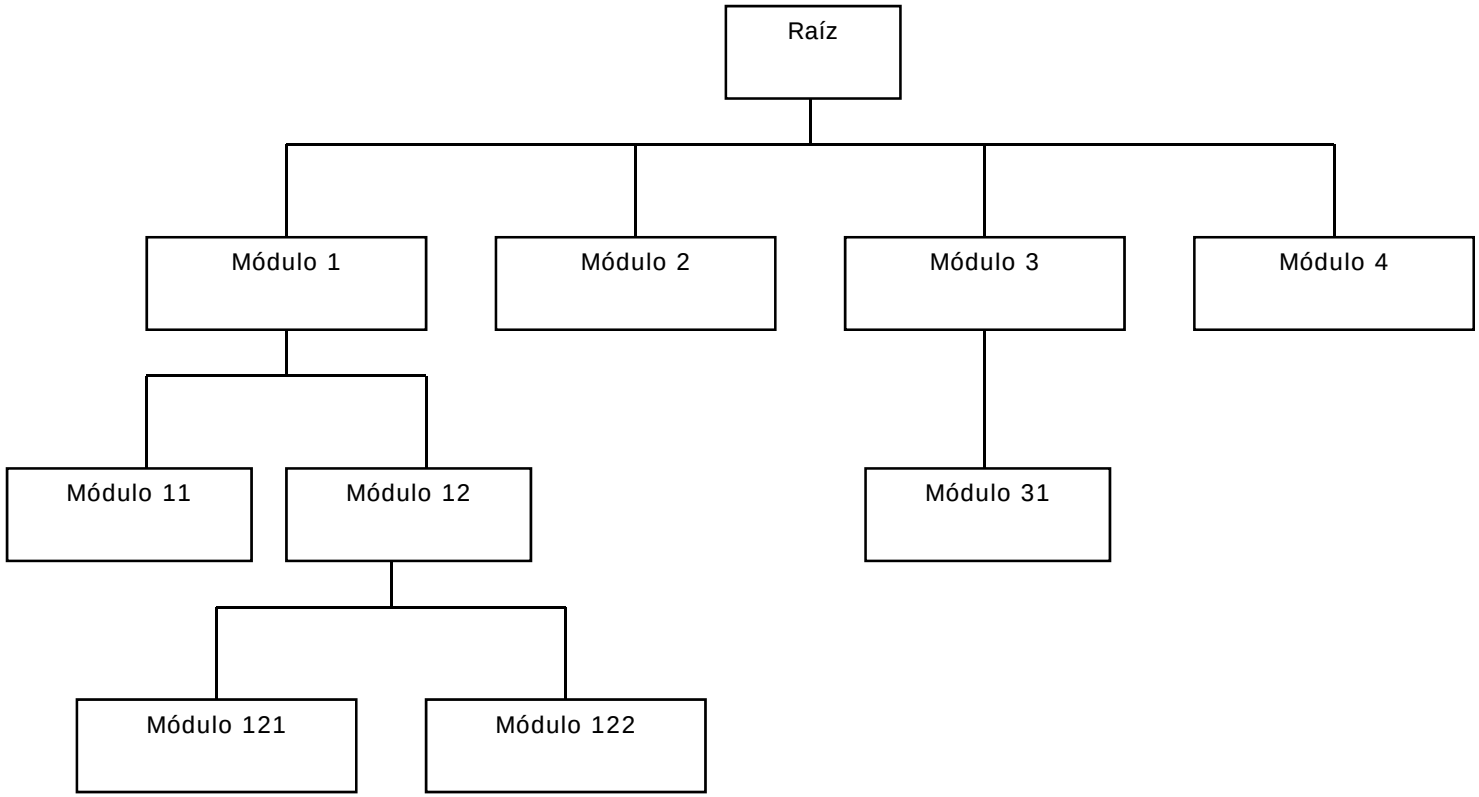
Unidad 4. Lógica de Programación

Prof. Eliana Guzmán U.

4.1 Programación Modular



4.2 Programación Estructurada

La programación estructurada significa escribir un programa de acuerdo a las siguientes reglas:

- El programa tiene un diseño modular.
- Los módulos son diseñados de modo descendente.
- Cada módulo se codifica utilizando únicamente, las tres estructuras básicas de control: secuencial, decisión y repetición.

4.3 Estructuras básicas de control

El estilo de escritura de los programas es una de las características más sobresalientes en las técnicas de programación.

La legibilidad de los algoritmos y posteriormente de los programas, exige que su diseño sea fácil de comprender y su flujo lógico fácil de seguir.

4.3 Estructuras básicas de control

- La programación modular enseña la descomposición de un programa en módulos más simples de programar, y
- La programación estructurada permite la escritura de programas fáciles de leer y modificar. En un programa estructurado, el flujo lógico se gobierna por las estructuras de control básicas:
  - Secuenciales.
  - Decisión (selección ó condición).
  - Repetición.

4.3.1 El flujo de control de un programa

El término **flujo de control** se refiere al orden en que se ejecutan las sentencias del programa. A menos que se especifique expresamente, el flujo normal de control de todos los programas es el **secuencial**. Este término significa que las sentencias se ejecutan en secuencia, una después de la otra, en el orden en que se sitúan dentro del programa.

4.3.1 El flujo de control de un programa

Uno de los más importantes avances luego de la aparición de los lenguajes de programación de alto nivel a finales de los años sesenta, fue el reconocimiento de que cualquier algoritmo, no importa su complejidad, podía ser construido utilizando combinaciones de tres estructuras de control de flujo estandarizadas (secuenciales, decisión y repetición) y una cuarta denominada invocación o salto.

4.3.1 El flujo de control de un programa

- Las sentencias de selección se utilizan para seleccionar cuáles sentencias se han de ejecutar a continuación. Y son: s**i** (if), **si-si_no** (if-else) y **según-sea** (case-of).
- Las sentencias de repetición o iterativas se utilizan para repetir un conjunto de sentencias y son: **mientras** (while), **repita-hasta** (repeat-until) y **desde** (for).

4.3.1 El flujo de control de un programa

- Las estructuras de decisión, repetición e invocación de subprogramas, permiten que el flujo secuencial del programa sea modificado en un modo preciso y definido con anterioridad.

4.3.1 El flujo de control de un programa

Hasta este momento todas las sentencias se ejecutaban secuencialmente en el orden en que estaban escritas en el código fuente, es decir, en ejecución secuencial.

Un programa escrito de esta forma siempre ejecutará exactamente las mismas instrucciones o acciones.

4.3.1 El flujo de control de un programa

Normalmente los programas necesitan alterar o modificar el flujo de control, así en la solución de muchos problemas se deben tomar acciones diferentes dependiendo del valor de los datos.

Ejemplos: cálculo de una superficie sólo si las medidas de los lados son positivas, la ejecución de una división solo si el divisor no es cero, etc.

4.3.1 El flujo de control de un programa

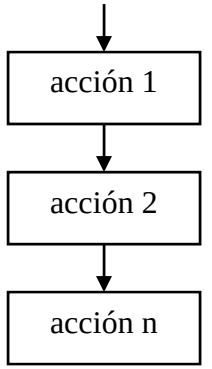
Una **bifurcación** (se llama también instrucciones de decisión, selección o de alternativas), es un punto del programa donde el flujo de control se divide.

Se utiliza para decidir o seleccionar qué instrucciones se van a ejecutar o si no se va a ejecutar nada, y continuar con el flujo de control normal del programa.

La decisión se realiza dependiendo de una **condición lógica** dada.

4.3.2 Estructura secuencial

Es aquella en la que una instrucción (acción) sigue a otra en secuencia. Las tareas se llevan a cabo de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el final del proceso. La estructura secuencial tiene una sola entrada y una sola salida.



4.3.3 Estructuras de decisión, selección o condición

- Los algoritmos tienen realmente utilidad cuando se requiere una descripción más detallada que una lista sencilla de instrucciones. Este es el caso cuando existen un número de posibles alternativas resultantes de la evaluación de una determinada condición lógica.
- Las estructuras de decisión se utilizan para tomar decisiones lógicas, de ahí que se suelan denominar también estructuras de selección o alternativas.

4.3.3 Estructuras de decisión, selección o condición

- En las estructuras de decisión se evalúa una condición lógica y en función del resultado de la misma, se ejecutará un conjunto de instrucciones u otro(s).
- Las condiciones se especifican usando expresiones lógicas. La representación de una estructura de decisión se hace con palabras en pseudocódigo (si, entonces, si_no), o con una figura en forma de rombo en los diagramas de flujo.

4.3.3 Estructuras de decisión, selección o condición

Las estructuras de decisión pueden ser:

- simples:
  - En pseudocódigo: **si-entonces**.
  - En Turbo Pascal: **if-then.**
- dobles:
  - En pseudocódigo: **si-entonces-sino.**
  - En Turbo Pascal: **if-then-else.**
- múltiples:
  - En pseudocódigo: **según-sea**.
  - En Turbo Pascal: **Case-of.**

4.3.3 Estructuras de decisión, selección o condición

Decisión simple (**si-entonces** / if-then)

- La estructura de decisión simple, **si-entonces**, ejecuta una determinada acción cuando se cumple una condición lógica. En esta estructura se evalúa la condición y
  - si la condición es *verdadera*, entonces ejecuta una o varias acciones.
  - si la condición es *falsa*, entonces no hace nada.

4.3.3 Estructuras de decisión, selección o condición

- Pseudocódigo:

si (condición) **entonces**
    <acción 1>
    <acción 2>
        .
        .
    <acción n>
fin_si

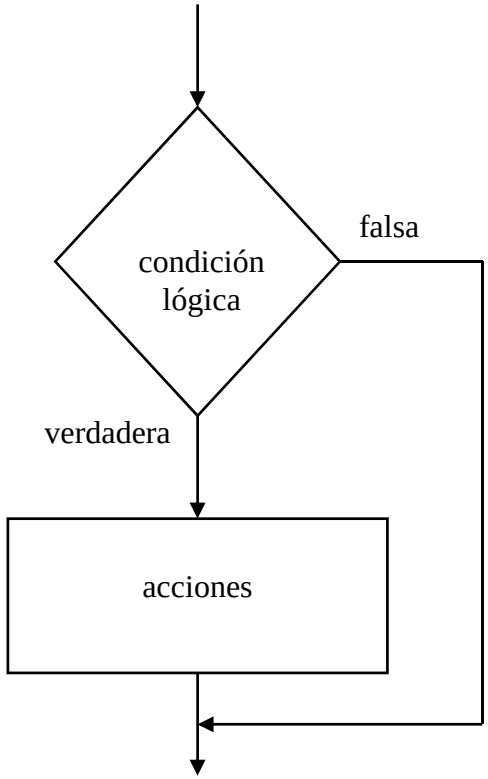
Turbo Pascal:

if (condición) **then**
  begin
      sentencias

  e**nd;**

4.3.3 Estructuras de decisión, selección o condición

- Representación en Diagrama de Flujo:



4.3.3 Estructuras de decisión, selección o condición

Decisión doble (**si-entonces-sino** / if-then-else)

- La estructura de decisión simple es muy limitada y normalmente se necesitará una estructura que permita elegir entre dos opciones o alternativas posibles, en función del cumplimiento o no de una determinada condición lógica. Si la condición lógica es verdadera se ejecuta un conjunto de acciones y si es falsa se ejecutará otro conjunto diferente de acciones.

4.3.3 Estructuras de decisión, selección o condición

Pseudocódigo:

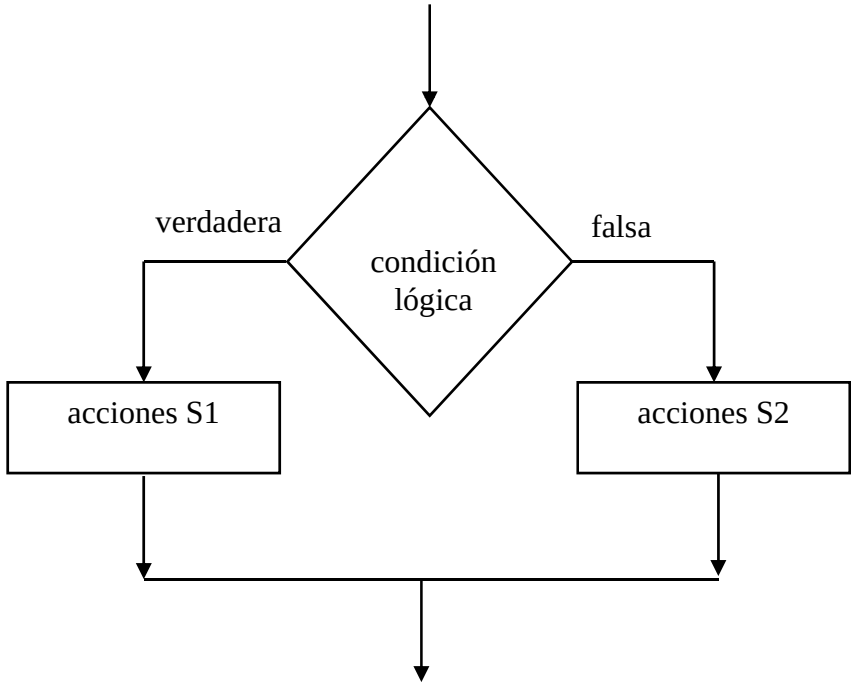
```
si (condición) entonces
   acciones S1
si_no
   acciones S2
fin_si
```

Turbo Pascal:

```
if (condición) then
  begin
    acciones S1
  end
else
  begin
    acciones S2
  end;
```

4.3.3 Estructuras de decisión, selección o condición

- Representación en Diagrama de Flujo:



4.3.3 Estructuras de decisión, selección o condición

Decisión múltiple (**según_sea**, case)

En la práctica es posible que existan más de dos alternativas posibles para elegir. Cuando se tienen muchas alternativas pueden existir serios problemas en la escritura y legibilidad de los algoritmos utilizando estructuras de decisión simples, dobles o anidadas.

4.3.3 Estructuras de decisión, selección o condición

La estructura de decisión múltiple evaluará una variable (tipo entero, caracter o lógica) que podrá tomar n valores distintos.

Según se elija uno de estos valores de la variable, se realizará una de las n acciones, o lo que es igual, el flujo de control del algoritmo seguirá un determinado camino entre los n posibles.

4.3.3 Estructuras de decisión, selección o condición

- Dentro de cada opción se pueden ejecutar una o varias acciones.
- Los valores que toma la variable E no tienen porque ser consecutivos ni únicos, se pueden considerar rangos de constantes numéricas o de caracteres (usando dos puntos. Ej: ‘A’ .. ‘Z’ ó 1..10).

Pseudocódigo:

según_sea E **hacer**
e1: acción S11
acción S12
.
.
acción S1a
e2: acción S21
acción S22
.
.
acción S2b
.
.
en: acción Sn1
acción Sn2
.
.
acción Snx
si_no
acción S
fin_según_sea

Pascal:

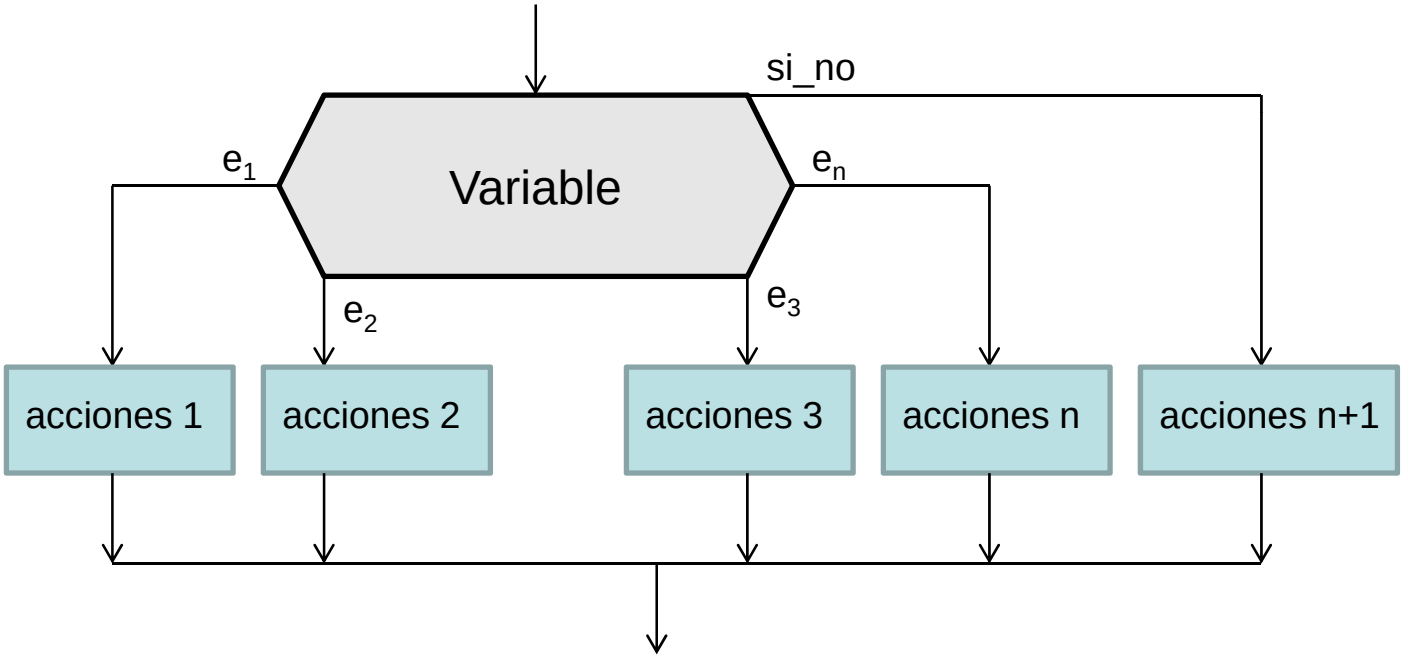
case E **of**
e1: acción1
e2: acción 2
end;

ó:

case E **of**
e1: acción1
e2: acción2
else
acción n
end;

donde:
E: es una variable de tipo entero, carácter o lógico.
acción: son instrucciones sencillas o compuestas (uso de begin y end)
ei: son los posibles valores o rango de valores que puede tomar la variable E.

Representación en Diagrama de Flujo:



4.3.4 Estructuras de decisión anidadas

Las estructuras de decisión simple (si-entonces) y doble (si-entonces-si_no), implican la selección de una de dos alternativas.

Es posible utilizar la instrucción **si** para diseñar estructuras de decisión que contengan más de dos alternativas. En este caso, una estructura si-entonces puede contener otra estructura si-entonces, y esta estructura puede contener a su vez a otra y así sucesivamente cualquier cantidad de veces, a su vez dentro de cada estructura pueden existir diferentes acciones.

Las estructuras **si** interiores a otras estructuras **si** se denominan anidadas.

4.3.4 Estructuras de decisión, anidadas

```
Si <condición lógica 1> entonces
    Si <condición lógica 2> entonces
       Acciones
    Fin_si
Fin_si
```

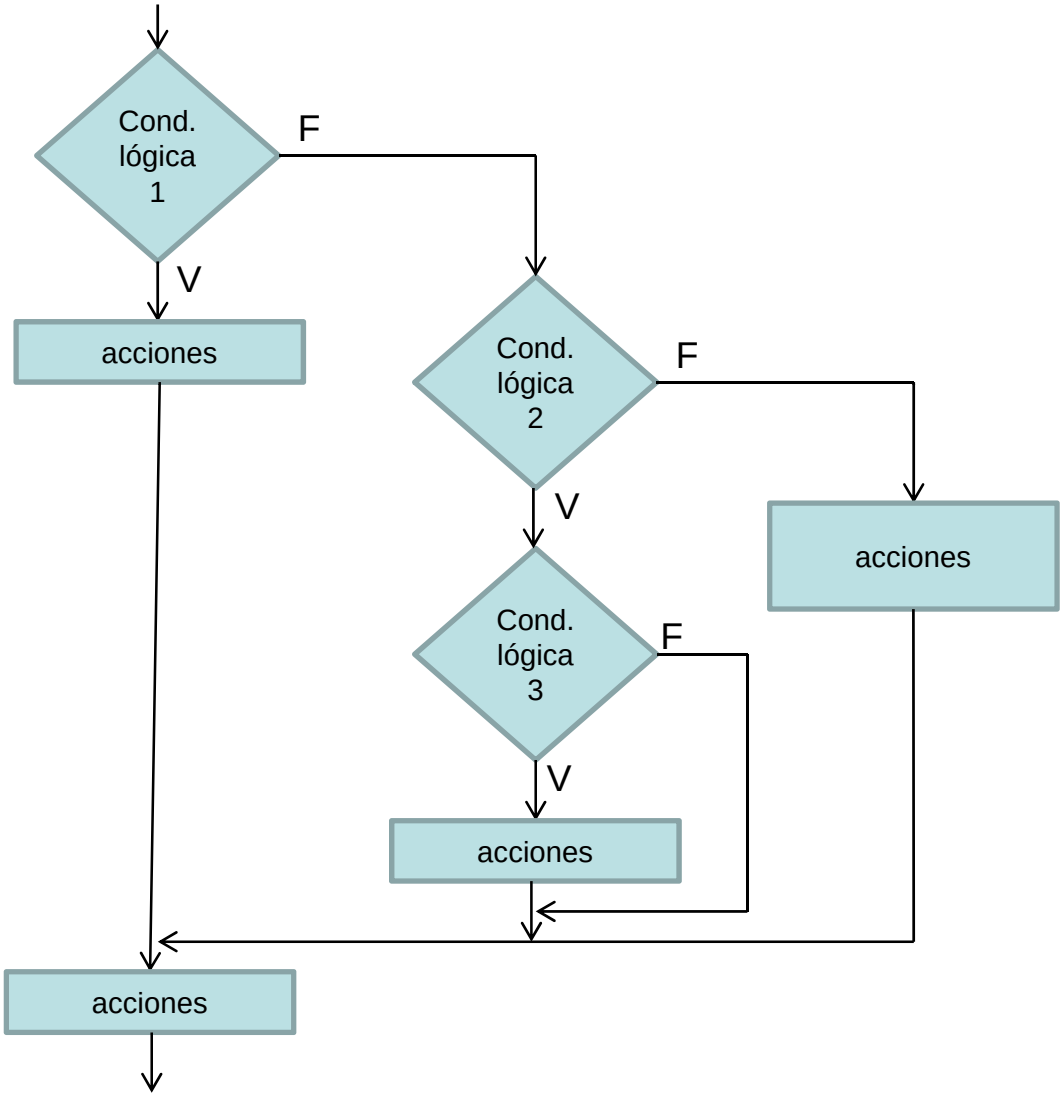
Se recomienda usar indexación para que el algoritmo sea más legible.

4.3.4 Estructuras de decisión, anidadas

Una estructura de selección de decisiones múltiples puede ser construida utilizando una estructura **si** con este formato:

```
Si <condición lógica 1> entonces
    <acciones>
si_no
   Si <condición lógica 2> entonces
      Acciones
   Si_no
      Si <condición lógica 3> entonces
        Acciones
      Si_no
       .
       .
      Fin_si
   Fin_si
Fin_si
```

Diagrama de Flujo de estructuras de decisión anidadas:



Ejemplos

Escriba los siguientes algoritmos que:

Ejemplo 1: Determine si un número es entero o no.

Ejemplo 2: Lea dos caracteres e indique si están en orden alfabético.

Ejemplo 3: Lea un carácter e indique si está o no comprendido entre las letras I y M ambas inclusive.

Ejemplo 4: permita emitir una factura, correspondiente a la compra de un artículo del que se adquieren una o varias unidades. El IVA es del 12% y si el monto bruto de la factura es mayor a Bs. 1000, se hará un descuento del 5%. Debe mostrar por pantalla los datos del cliente que exige el SENIAT (Nombre, CI y dirección).

Ejemplo 5: Escriba los nombres de los días de la semana en función del valor de una variable DIA introducida por teclado (usando una estructura de decisión múltiple).

Ejemplos

- Ejemplo 6: Convierta las calificaciones alfabéticas A, B, C, D, y E a calificaciones numéricas 8, 7, 6, 5 y 4 respectivamente (usando una estructura de decisión múltiple).
- Ejemplo 7: Determine el precio del pasaje de ida y vuelta en ferrocarril, el cual depende de la distancia a recorrer y del número de días de estancia. Si la cantidad de días es superior a 7 y la distancia superior a 800 km el billete tiene un descuento del 30 por ciento. El precio por km es de Bs. 50.

  Además, a las personas de la tercera edad (65 o más años), se les hará un descuento del 8%.

Ejemplos

Ejemplo 8: Un ángulo se considera agudo si es menor de 90 grados, obtuso si es mayor de 90 grados y recto si es igual a 90 grados. Utilizando esta información, escribir un algoritmo que acepte un ángulo en grados e indique el tipo de ángulo correspondiente a los grados introducidos.

Ejemplo 9: Los empleados de una fábrica trabajan en dos turnos: diurno y nocturno. Se desea calcular el salario diario de acuerdo a las siguientes condiciones:

a. la tarifa de las horas diurnas es de Bs. 20.

b. la tarifa de las horas nocturnas es de Bs. 35.

c. los días domingo, la tarifa se incrementará en Bs. 10 el turno diurno y Bs. 18 el turno nocturno.

Ejemplos

- Ejemplo 10: resuelva una ecuación de 2do. Grado, incluyendo el caso de raíces imaginarias.
- Ejemplo 11: Lea tres números X, Y, Z y muestre en pantalla el mayor valor, suponiendo que los tres valores son diferentes.
- Ejemplo 12: seleccione la operación aritmética a ejecutar entre dos números dependiendo del valor de una variable denominada selecc_op de tipo caracter.
- Ejemplo 13: determine si tres puntos en el plano XY forman un triángulo.

4.3.4 Estructuras de Repetición

- Las computadoras están especialmente diseñadas para todas aquellas aplicaciones que requieren una operación o conjunto de ellas que deban repetirse varias veces.
- Un tipo muy importante de estructura, es el algoritmo necesario para repetir una o varias acciones, un número determinado de veces.

4.3.4 Estructuras de Repetición

- Bucles: son las estructuras que repiten una secuencia de instrucciones, una cantidad determinada de veces.

- Iteración: es la acción de repetir una vez, la ejecución de una instrucción.

4.3.4 Estructuras de Repetición

Las dos principales preguntas a realizarse en el diseño de un bucle son:

1. **¿qué instrucciones se quieren repetir?**
2. **¿cuántas veces se deben repetir?**

4.3.4 Estructuras de Repetición

Para detener la ejecución de los bucles se utiliza una condición lógica de parada.

Dicha condición puede estar al final o al principio del bucle, y siempre debe existir en el interior del bucle una instrucción que modifique dicha condición en cada repetición que se realice.

Tipos de estructuras de repetición

Las estructuras de repetición pueden ser de dos tipos:

1. Condicionales:
    1. Mientras (while).
    2. Repetir (repeat).
2. Indexadas o indizadas:
    1. Desde (for).

Estructuras de repetición condicionales

Las estructuras de repetición **condicionales**, dependen de la evaluación de una condición lógica de parada. La cual se evalúa tan pronto se encuentra en el algoritmo, puede estar al inicio de la estructura (mientras) o al final (repetir).

Estructuras de repetición indexadas o indizadas

Las estructuras de repetición indexadas o indizadas, dependen de un índice, el cual es una variable tipo entero que controla la cantidad de veces que se repite el bucle.

Para usarlas se debe conocer de antemano, la cantidad de veces que se quiere repetir el bucle.

Estructura de Repetición: Mientras

- Es aquella en la que el cuerpo del bucle se repite, siempre y cuando, se evalúe la condición lógica de parada como verdadera.
- En esta estructura la condición lógica de parada se encuentra al inicio.

Estructura de Repetición: Mientras

- Cuando se ejecuta la estructura mientras, lo primero que sucede es que se evalúa la condición lógica, si es falsa no se toma ninguna acción y el programa continúa ejecutando la siguiente instrucción, pero si es verdadera, se ejecuta el cuerpo del bucle, después de lo cual se evalúa de nuevo la condición lógica. Este proceso de ejecuta una y otra vez mientras la condición lógica sea verdadera.

Estructura de Repetición: Mientras

- En resumen, se repite el bucle cuando la condición lógica es verdadera, por lo tanto, finaliza la repetición cuando la condición se hace falsa.

Estructura de Repetición: Mientras

Pseudocódigo:

mientras (condición lógica) **hacer**

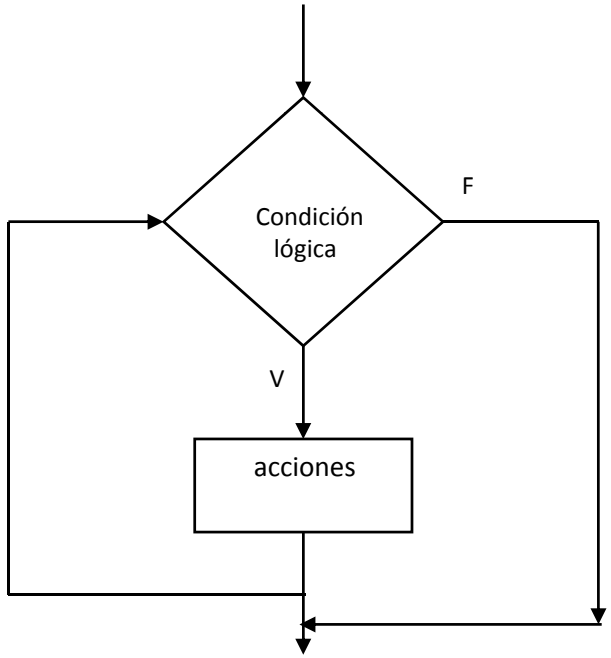
acción 1
acción 2
.
.
acción n

Bucle

fin_mientras

Estructura de Repetición: Mientras

Diagrama de flujo:



Estructura de Repetición: Mientras

Una estructura de repetición mientras puede generar un:

- Bucle de Repetición Cero:

Como en esta estructura lo primero que se hace es la evaluación de la condición lógica, si se evalúa falsa, entonces el cuerpo del bucle nunca se ejecuta.

Estructura de Repetición: Mientras

Bucle infinito:

- Algunos bucles no finalizan la repetición, por error en su diseño. Esto quiere decir que un bucle no se termina nunca, porque la condición lógica siempre se evalúa como verdadera y no se detiene su ejecución.
- Un bucle que nunca se termina se denomina bucle infinito o sin fin. Los bucles sin fin no intencionados son perjudiciales para la programación y deben evitarse siempre.

Estructura de Repetición: Mientras

Por ejemplo, el siguiente bucle que visualiza el interés producido por un capital a las tasas de interés comprendidas entre 10% y 20%.

leer (capital)

tasa ← 10

mientras (tasa <> 20) **hacer**

  interés ← (tasa*capital)/100

  escribir(‘interés producido’, interés)

  tasa ← tasa + 2

fin_mientras

escribir(capital)

Estructura de Repetición: Mientras

Los valores sucesivos de la variable tasa serán 10, 12, 14, 16, 18, 20 de modo que al tomar tasa el valor 20 se detendrá el bucle y se escribirá el mensaje continuación. Si se usa la siguiente sentencia:

$$tasa \leftarrow tasa + 3$$

La variable tasa nunca tomará el valor de 20 y el bucle sería infinito, la sentencia correcta para terminar el bucle sería: tasa < 20 ó tasa <= 20

Estructura de Repetición: Mientras

Regla Práctica:

Las condiciones lógicas en las estructuras de repetición, es conveniente que sean del tipo mayor que o menor que, en lugar de pruebas de igualdad o desigualdad.

Estructura de Repetición: Mientras

Formas de terminar la repetición de un bucle:

Si el algoritmo está leyendo una lista de valores con un bucle mientras, se debe incluir algún tipo de mecanismo para terminar el bucle. Existen tres métodos típicos para terminar un bucle de entrada de datos:

Estructura de Repetición: Mientras

1. Conocer de antemano la cantidad de veces que se va a repetir el bucle (cantidad de datos de entrada).
2. Preguntar antes de cada repetición.
3. Usar un valor centinela de entrada.

1. Conocer de antemano la cantidad de veces que se va a repetir el bucle (cantidad de datos de entrada).

Pseudocódigo:

```
escribir('Introduzca la cantidad de números que desea sumar: ')
leer(n)
total ← n
suma ← 0
mientras (total > 0) hacer
   escribir ('Introduzca el número que desea sumar: ')
   leer(numero)
   suma ← suma + numero
   total ← total - 1
fin_mientras
escribir('La suma de los ',N, 'números es: ',suma)
```

2. Preguntar antes de cada repetición: este método no es el más apropiado para gran cantidad de datos de entrada, pero es útil cuando no se conoce la cantidad de veces que desea repetir el bucle.

Pseudocódigo:

```
Suma ← 0
resp ← 's'
mientras (resp='S') ó (resp='s') hacer
  escribir('Introduzca un número: ')
  leer(numero)
  suma ← suma + numero
  escribir('Existen más números s/n: ')
  leer(resp)
fin_mientras
```

Este método a veces es aceptable y es muy útil en ciertas ocasiones, pero suele ser tedioso para listas grandes, es este caso es preferible incluir una señal de parada.

3. Usar un valor centinela de entrada: es el método más correcto para terminar un bucle que lee una lista de valores.

Un valor centinela es un valor especial usado para indicar el final de una lista de datos.

Un valor centinela no es un valor válido de entrada, por eso termina la ejecución del bucle.

Si la lista de datos son números positivos, un valor centinela puede ser un número negativo que indique el final de la lista:

Pseudocódigo:

```
suma ← 0
escribir(‘Introduzca un número: ‘)
leer(numero)
mientras (numero >= 0) hacer
   suma ← suma + numero
   escribir(‘Introduzca un número: ‘)
   leer(numero)
fin_mientras
escribir(suma)
```

Observe que el último número leído de la lista no se añade a la suma si es negativo, ya que se sale del bucle, además cuando se usa un valor centinela, éste debe leerse al final del bucle y se invierte el orden de las instrucciones de lectura y suma para evitar realizar la suma del valor centinela.

Ejemplos

1. Un profesor de cálculo 10 tiene las notas del primer parcial de los n estudiantes que presentaron dicho examen. Escriba un algoritmo que le permita a este profesor conocer:
   a) La cantidad y el porcentaje de estudiantes que aprobaron el examen.
   b) La cantidad de estudiantes que sacaron más de 16 puntos.
   c) La mayor nota obtenida en este examen.
   d) El promedio de las notas obtenidas en este examen.

2. Escriba un algoritmo que calcule cuántos años tarda en duplicarse un capital depositado en un banco al 5% de interés anual.

3. Escriba un algoritmo que calcule el aumento de sueldo mensual, de cada uno de los empleados de una empresa y el total de la nómina de ese mes con el aumento.

   Si un empleado gana menos de Bs. 2600 el aumento será del 15% y si gana Bs. 2600 ó más el aumento es del 12%.

   Debe mostrar por pantalla el nombre y el sueldo con aumento de cada trabajador.

4. En la aduana del aeropuerto de Maiquetía, se quiere tener un sistema de control para un vuelo que aterriza con pasajeros. Escriba un algoritmo que le indique al agente de aduana:

a) La cantidad de extranjeros que viajaban en dicho vuelo.
b) La edad promedio de los pasajeros extranjeros y la de los venezolanos.
c) La cantidad de niños (menores de 12 años) que viajaban en dicho vuelo.
d) La cantidad de pasajeros que viajaban con una mascota.

5. Se realiza una encuesta a n personas que viven en la ciudad de Mérida. Escriba un algoritmo que le permita determinar:

a) El peso y estatura promedio de la muestra.
b) El peso promedio de las mujeres encuestadas.
c) Cantidad de hombres que miden 1.70 m. y tienen un peso mayor a 80 Kgr.
d) El menor registro de estatura de la muestra.

Hasta aquí es la materia del 2do examen parcial.

Estructura de Repetición: Repetir

En la estructura mientras si el valor de la condición lógica es inicialmente falso, el cuerpo del bucle no se ejecutará, por ello se necesitan otros tipos de estructuras de repetición.

Existen muchas situaciones en las que se desea que un bucle se ejecute al menos una vez, antes de comprobar la condición lógica de repetición.

Estructura de Repetición: Repetir

La estructura repetir (repeat) se ejecuta hasta que se cumpla una condición determinada que se comprueba al final del bucle.

El bucle repetir-hasta_que, se repite cuando la condición lógica se evalúa como falsa, justo lo opuesto a la sentencia mientras, por lo tanto, se detiene la repetición cuando la condición lógica se hace verdadera.

Estructura de Repetición: Repetir

Pseudocódigo:

repetir

acción 1

acción 2

.

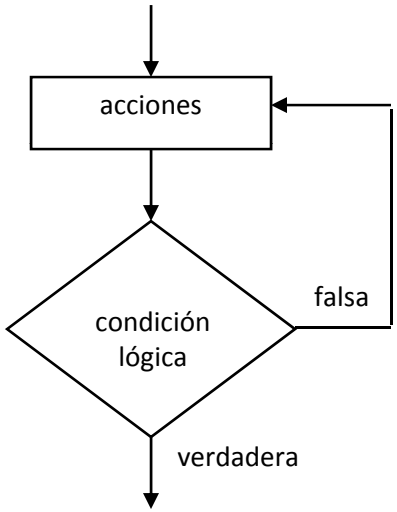
.

acción n

hasta_que (condición lógica)

Estructura de Repetición: Repetir

Diagrama de flujo:



Con una estructura repetir el cuerpo del bucle se ejecuta siempre por lo menos una vez, a continuación se evalúa la condición lógica, si es falsa el cuerpo del bucle se repite, se evalúa de nuevo la condición, si es verdadera el bucle termina y el programa sigue en la siguiente instrucción.

Diferencias entre las estructuras mientras y repetir

- En la estructura mientras la condición lógica se evalúa al inicio, en la estructura repetir dicha condición se encuentra al final.
- La estructura mientras termina cuando la condición es falsa, en cambio la estructura repetir termina cuando la condición es verdadera.

Diferencias entre las estructuras mientras y repetir

- En la estructura repetir el cuerpo del bucle se ejecuta siempre al menos una vez, por el contrario, la estructura mientras es más general y permite la posibilidad de que el bucle no se ejecute nunca.
- Para usar la estructura repetir tiene que estar seguro de que el cuerpo del bucle, bajo cualquier circunstancia, se ejecutará al menos una vez.

Ejemplo 1: Escribir los números del 1 al 100.

```
algoritmo escribir_numeros
var
  entero: num
inicio
  num ← 1
  repetir
      escribir(num)
      num ← num + 1
  hasta_que num > 100
fin
```

Ejemplo 2: Validación de entrada de datos. Leer los números correspondientes a los meses del año, entradas válidas 1-12, rechazar cualquier otra entrada.

```
algoritmo validar_mes
var
  entero: mes
inicio
  repetir
    escribir('Introduzca el número correspondiente a un mes: ')
    leer(mes)
    si (mes < 1) o (mes > 12) entonces
       escribir('Introdujo un valor que no es válido para un mes')
    fin_si
  hasta_que (mes >= 1) y (mes <= 12)
fin
```

Estructura de Repetición: Desde

Esta estructura de repetición, es del tipo indexada o indizada.

En muchas ocasiones se conoce de antemano el número de veces que se desean ejecutar las acciones de un bucle. En estos casos, en el que el número de iteraciones es fijo, se debe usar la estructura desde ó para (for).

Estructura de Repetición: Desde

La estructura desde ejecuta las acciones del cuerpo del bucle un número dado de veces, y de modo automático controla el número de iteraciones o pasos a través del cuerpo del bucle.

Estructura de Repetición: Desde

La estructura desde comienza con un valor inicial de la variable índice y las acciones del bucle se ejecutan, a menos que el valor inicial sea mayor que el valor final.

La variable índice se incrementa en uno y si este nuevo valor no excede al final, se ejecutan de nuevo las acciones. Por consiguiente, las acciones específicas en el bucle se ejecutan para cada valor de la variable índice desde el valor inicial hasta el valor final con el incremento de uno en uno.

Estructura de Repetición: Desde

Pseudocódigo:

desde i ← vi **hasta** vf **hacer**

  acciones

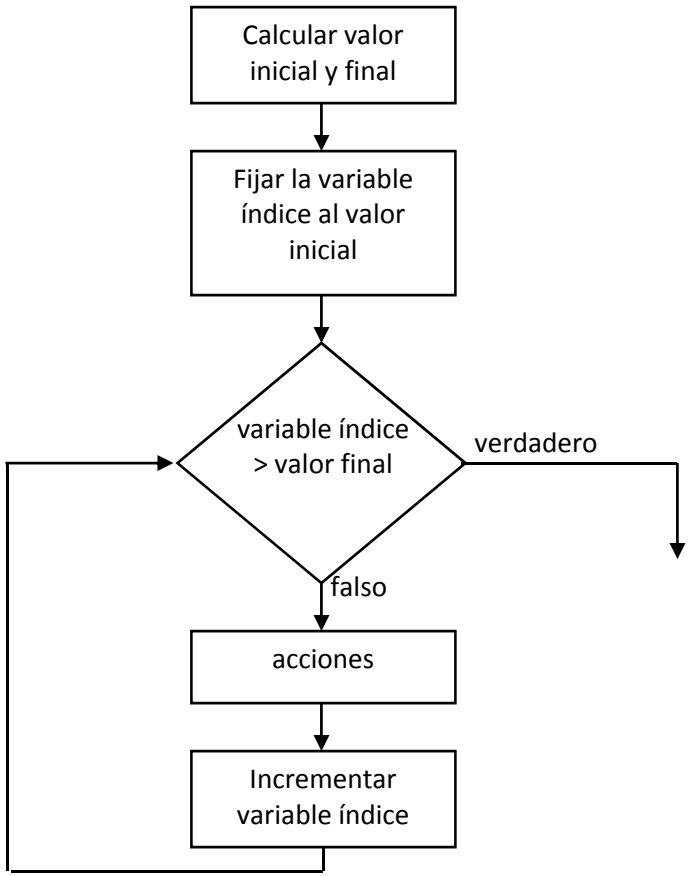
  .

  .

fin_desde

Estructura de Repetición: Desde

Diagrama de flujo:



Estructura de Repetición: Desde

El incremento de la variable índice siempre es 1 si no se indica expresamente lo contrario.

Turbo Pascal solo admite incrementos de 1: tanto negativos como positivos.

La variable índice o de control normalmente será de tipo entero y es normal emplear letras como I, J, K.

Estructura de Repetición: Desde

Si el valor inicial de la variable índice es menor que el valor final, los incrementos deben ser positivos, ya que en caso contrario la secuencia de acciones no se ejecutaría.

De igual modo si el valor inicial es mayor que el valor final, el incremento es negativo, es decir un decremento.

Estructura de Repetición: Desde

desde i ← 20 **hasta** 10 **hacer**
  acciones
fin_desde

desde i ← 20 **hasta** 10 **decremento** 1 **hacer**
  acciones
fin_desde

El pseudocódigo de la izquierda, no se ejecutaría ya que el valor inicial es mayor que el valor final, se supone un incremento positivo de 1 y se producirá un error. El pseudocódigo correcto sería el de la derecha.

Estructuras de repetición anidadas

Las reglas para construir estructuras de repetición anidadas son:

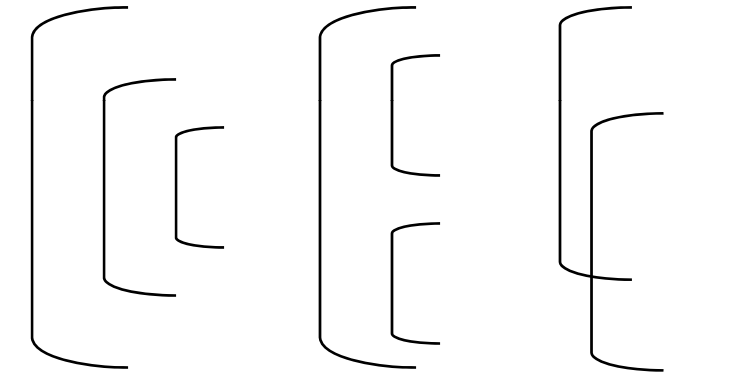
- La estructura interna debe estar incluida totalmente dentro de la estructura externa y,
- No puede existir solapamiento.

Estructuras de repetición anidadas

Las variables índice o de control de los bucles, toman valores de modo tal que por cada valor de la variable índice del ciclo externo, se debe ejecutar totalmente el bucle interno.

Es posible anidar cualquier tipo de estructura repetitiva, siempre y cuando, se cumplan las 2 reglas anteriores.

Estructuras de repetición anidadas



Correcto Correcto Incorrecto