Contenido

SENTENCIAS DE CONTROL

Son las estructuras básicas necesarias para organizar el flujo de control en un programa. Tres de éstas son: la secuencial, la de alternativa y de repetición, que constituyen el fundamento de la organización necesaria para respaldar un proceso sistemático de programación.

Esto permitirá al programador tener programas con una mejor organización lo cual le permitirá elaborar programas más fáciles de escribir, leer y modificar.

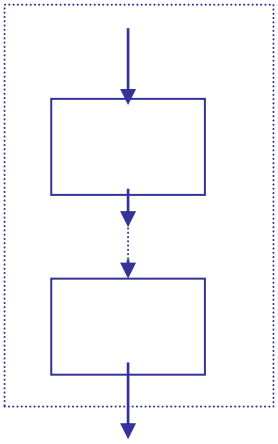
3.1 ESTRUCTURA SECUENCIAL

Conjunto de instrucciones que se ejecutan en forma consecutiva, es decir una primero, luego la siguiente sentencia y así hasta llegar a la última instrucción. Es muy útil para problemas sencillos pero no en problemas que requieren la toma de decisiones y realizar procesos masivos de manejo de datos.

Formato:

INSTRUCCIÓN 1
INSTRUCCIÓN 2
.
.
.
INSTRUCCIÓN n

DIAGRAMA



Donde en cada símbolo de proceso se escriben la(s) instrucción(es).

Ejemplo # 1:

```
sum = a + b + c
prom =  sum/2
Imprimir  (“valor de la suma =”, sum)
Imprimir (“valor promedio =”, prom)
```

3.2. ESTRUCTURA ALTERNATIVA

Controla la ejecución de uno o varios bloques de instrucciones, dependiendo de si se cumple o no alguna condición.

Una **Condición** es una expresión lógica, es decir, una combinación de variables y/o constantes u expresiones aritméticas, que usa operadores relacionales y/o lógicos los cuales producen resultados ciertos o falsos.

3.3 OPERADORES RELACIONALES

Relacionan un término con otro para definir su igualdad, jerarquía o cualquier otra relación que guarden.

Operador	Propósito	Ejemplo
<	Menor que	A < 5
<=	Menor o igual a	A <= 5
>	Mayor que	A > 5
>=	Mayor o igual a	A >= 5
==	Igual a	A == 5
<>	No igual a , distinto de	A < > 5

- Siempre se devuelve un valor booleano que puede ser C(cierto) ó F(falso)

Ejemplos :

entero a,b
flotante x = 15.0 y = 18.0 z = 20.0

a) x ==y → F

b) x < y → C

c) y <= z → F

OPERADORES LÓGICOS

Un operador lógico permite asociar una o más expresiones relacionales. Si tenemos más de una expresión relacional, ésta se conoce como condición compuesta, porque estarán relacionadas a través de operadores lógicos.

Operador	Propósito
Y	AND (Y), da como resultado cierto, si ambas comparaciones son ciertas
O	OR (O), Si una de las comparaciones es C (cierta) , el resultado es cierto
!	NOT, da 0, si el resultado es 1 da 1, si el resultado es 0

Reglas:

- Los operandos pueden ser de cualquier tipo.
- El resultado es de tipo booleano.

Ejemplo 1. entero p, q
booleano p
flotante x, y, z
x=15 y= 18 z= 20
p= x==y **/* resultado F */**
q= (x< y) Y (y<=z) /* resultado C */

Ejemplo 2:
p=10
q=0
! (p < q) -→ /* resultado F */

Ejemplo 2:
entero 1 = 7
flotante f = 5.5
caracter c = 'w'
(i >= 6) Y (c==´w´) → C

Jerarquía de los operadores relacionales y lógicos

1) !
2) <, <=, >, >=
3) ==, < >
4) Y
5) O

Ejemplo:

(10>5) Y !(10<9) O (3<=4)
C F C
F
C
// resultado es cierto

3.3 ESTRUCTURAS DE ALTERNATIVA

Son aquellas sentencias que trabajan con los operadores lógicos y/o relacionales, y que al ser evaluadas solo pueden dar una respuesta de cierto o de falso.

Existen tres tipos de alternativas básicas:

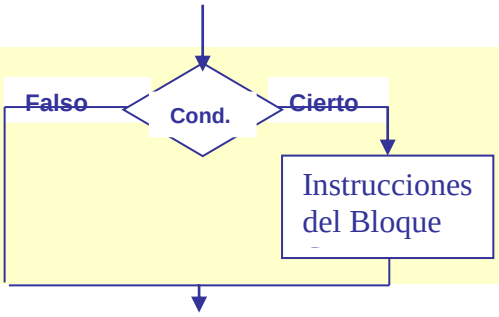
1. Alternativa Simple
2. Alternativa Doble
3. Alternativa Múltiple

a.Alternativa Simple

Evalúa sólo por la parte cierta de una condición.

Formato

Si (condición) **entonces**
{
sentencia(s)
sentencia(s)
}
Fin Si



a.
```
entero i
Imprimir("Introduzca un número")
Leer(estado)
 Si(i < 1000)  O (estado == 'R'))
              Imprimir( i)
 Fin Si
```

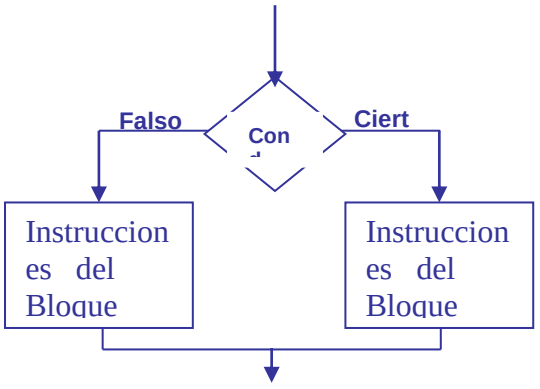
b.
```
Si (x<0) entonces
   Imprimir ( x)
Fin si
```

c.
```
Si (x <= 30) entonces
{
   y = 3 * 6 + c
   Imprimir("%f", y)
}
Fin Si
```

b. Alternativa Doble

Realiza una comparación y a continuación ejecuta una de dos acciones dependiendo del resultado (verdadero o falso) de una condición.

Formato:
```
Si (condición) entonces
   sentencia(s)
De otro modo
   sentencia(s)
Fin si
```



Si se presenta **más de una** instrucción (sentencia) por el bloque cierto (que la condición se cumpla) o por el bloque falso que la condición no se cumpla), deben ponerse las instrucciones entre paréntesis { }.

Ejemplos :

```
Si (x==7) entonces
        Imprimir(" Es igual")
  De otro modo
        Imprimir("Desigual")
Fin si
```

```
Si  (balance >0) entonces
 {
   Imprimir("alerta",noc)
   credito = 0
 }
De otro Modo
    credito = 100.0
Fin si
```

```
Si ( x<= 3) entonces
      y = 3 * (x * x)
  De otro modo
    y = 2 * ((x – 3) +2)
Fin Si
Imprimir(y)
```

c. Alternativa Múltiple Si De Otro ModoSi De otroModo

Permite evaluar más de una expresión de comparación, pero sólo ejecuta una de ellas. Si la primera condición no se cumple, se compara la segunda y así sucesivamente. En el caso de que no se cumpla ninguna de las comparaciones se ejecutan las sentencias correspondientes al ***De otro modo***.

Formato:

```
Si (expresión) Entonces{
    Grupo de sentencias   }
De otroModo Si (expresión) Entonces {
     Grupo2 de sentencias    }
De otro Modo Si(expresión) entonces {
    Grupo3 de sentencias  }
De otro Modo{
    Grupo_n de sentencias}
Fin si
```

Ejemplo :

```
entero x
Leer(x)
Si ( (x>3) Y (x<4))
            Imprimir(“Entre tres y cuatro ”)
De otro Modo Si   ((x>6) Y (x<8)) Entonces
              Imprimir(“Entre seis y ocho”)
 De otro Modo
          Imprimir(“Restantes casos”)
Fin Si
```

3.4 SENTENCIAS DE REPETICIÓN

Son aquellas que controlan la repetición de un conjunto de instrucciones mediante la evaluación de una condición.

Toda estructura de repetición está compuesta por los siguientes elementos:

1. **Condición**: expresión a evaluar (define la entrada o no al ciclo).
2. **Bloque de instrucciones:** instrucciones a ejecutarse.
3. **Ciclo o iteración**: son las veces que se ejecuta el ciclo.

Las sentencias de control repetitivas que veremos a nivel de algoritmo son:

- Mientras
- Para

Es necesario definir procesos para controlar la condición de un ciclo, se conozca o no el límite de iteraciones. Por eso éstos se definen como:

a. **Ciclos Definidos:** Son aquellos donde se conoce el número de veces que es necesario ejecutar el ciclo.

a.1 **Ciclo controlado por Contador**

Requiere:

1. nombre de la variable de control o contador
2. valor inicial de la variable de control
3. incremento o decremento
4. establecer la condición que compruebe el valor final de la variable de control.

b. **Ciclos no Definidos:**

Son aquellos donde no se conoce con anticipación el número de veces que se ejecutará el ciclo. Se utilizan diferentes técnicas como: ciclo controlado por valor centinela, ciclo controlado por respuesta, etc.

b.1 **Ciclo controlado por respuesta:** se utiliza en aquellas ocasiones en que se le da al usuario la opción de terminar el ciclo.

Requiere:

a. el nombre de la variable de control
b. valor inicial de la variable de control
c. la condición que comprueba el valor final de la variable de control
d. Colocar dentro del bloque de instrucciones la solicitud de la respuesta del usuario, para la variable de control.

b.2 **ciclo controlado por valor centinela:** se utiliza en aquellas ocasiones en que no se conoce el número de veces que el ciclo se ejecuta. El valor centinela es aquel que termina el ciclo, dándole un valor diferente a un campo que forma parte del conjunto de datos de entrada.

Requiere:

a. el nombre de la variable de control
b. la condición que comprueba el valor final de la variable de control
c. lectura de la variable de control antes de entrar al ciclo
d. lectura de la variable de control dentro del bloque de instrucciones

Para realizar procesos con ciclos definidos, las estructuras de repetición requieren de un contador, que debe ser inicializado.

Contador: variable que se utiliza para contar las veces que se ejecuta el ciclo un **número constante** de veces.

Formato :

Variable contador = variable contador +/- [constante numérica]

Ejemplo :

```
Cont = cont + 2
Cont = 5
Cont = cont – 1
```

Reglas :

1. **Declarar** la variable contador
2. **Inicializar** con un valor que puede ser 0 o cualquier otro antes de empezar el ciclo.

   int cont cont = 0
3. **Incrementar** con valor constante

```
cont = cont + 1
```

En estos proceso también se podrían realizar procesos como: acumular el dinero de una empresa, calcular el total de ventas de un almacén, etc. Para ello se utiliza un tipo de variable conocida como acumulador.

Acumulador: es una variable cuya función es almacenar **cantidades variables**, resultantes de sumas sucesivas.

Formato:

Variable acumulador = variable acumulador + variable

Ejemplo:

```
flotante salariobruto
flotante  acum.
acum. = 0
-------
Leer(salariobruto)
acum. = acum. + salariobruto
```

SENTENCIA MIENTRAS:

Ejecuta una sentencia, simple o compuesta, cero o más veces mientras lacondición sea cierta.

Formato: **Mientras** (expresión)
Sentencia 1
Sentencia 2
Sentencia n
Fin Mientras

La ejecución sucede si:

1. se evalúa la condición
2. si el resultado es Falso, no se ejecuta la(s) sentencia(s) y se pasa a ejecutar la siguiente instrucción.
3. si el resultado es Cierto se ejecutan las instrucciones del Mientras y se repite el proceso.

Ejemplo 1: Imprimir los 10 primeros números

```
entero c ← contador
c = 0 ← Inicializar  contador
Mientras (c < 10)   ← condición
   c =  c + 1 ←-- incremento
    Imprimir("Valor de c =", c)
Fin Mientras
```

Ejemplo 2: Usando Ciclo con Respuesta

Ejemplo: Leer un grupo de números y sumarlos. **Usar ciclo con respuesta.**

```
Entero num,  suma
caracter resp ← definición de variable de control
suma=0

resp = 's' ←inicializar variable de control
Mientras(( resp =='s') O (resp =='S')) ← condición
  Imprimir("Ingrese un número")
  Leer(num)
  suma = suma + num
  Imprimir("Desea leer otro número? s/n")
  Leer(resp)  ← leer respuesta
Fin Mientras
Imprimir("La suma es ",suma)
```

Ejemplo: **L**eer una serie de velocidades siempre y cuando la velocidad sea diferente de -9999. **Usando valor Centinela**

```
 c=0
Imprimir("Ingrese la velocidad") ← 1ª lectura
 Leer(vel)

suma=0

Mientras( vel <> -9999)  ← condición

   suma = suma + vel
   c = c + 1
   Imprimir("Ingrese la velocidad") ← leer valor
   Leer(vel)
 Fin Mientras
 prom = sum/cont
```

SENTENCIA PARA

Permite ejecutar una sentencia simple o compuesta repetidamente un número conocido de veces. Para utilizarla se debe conocer de antemano el número de veces a iterar.

Formato:

```
Para variable de control = Valor Inicial, Valor Final, Incremento/decremento
    sentencia(s)
Fin Para
```

Variable de control = variable de tipo entero, que se incrementa o decrementa en un valor fijo

Valor Inicial, valor final: puede ser una constante o el valor de una variable que determinan el inicio y fin del algoritmo. Deben ser de tipo entero.

Incremento/decremento: constante de tipo entero, que define la cantidad en que será incrementada o decrementada la variable de control.

Cuando se ejecuta la sentencia Para:
la condición se evalúa y se comprueba antes de entrar al ciclo, el incremento es evaluado al final del ciclo.

1. valor inicial se asigna a variable de control

2. Se valida, si el valor de la variable de control es menor o mayor que valor final, como se establece esto, si variable de control es menor que valor final la condición será de menor o igual a valor final, ahora si variable de control es mayor que valor final, la condición seria si variable de control es mayor o igual que valor final.

3. Si la condición se cumple, se ejecuta el bloque de instrucciones y al llegar a fin para, se incrementa/decrementa según sea el caso a la variable de control y se vuelve al paso 2.

Caso 1: Sentencia simple. Se imprimen los números del 1 al 100

```
------
Entero x
Para x = 1, 10, 1 ← inicialización /Valor Final/incremento
        Imprimir( x)
    Fin Para
```

Caso 2: presentación de un ejemplo con sentencias compuestas.

```
----
Entero k , kk
  Para k = 7 ,122 ,7
      imprimir("Se imprime el valor de k:",k)
       kk = k *k
       Imprimir("Cuadrado de  k: ",kk)
        x=x+1
       Imprimir("número de veces ejecutado:",x)
   Fin Para
```

a. Bucles Anidados

Son aquellos donde tenemos una sentencia Para colocada dentro de otra sentencia Para. El bucle interno se ejecutará totalmente por cada valor del bucle que lo contiene.

Ejemplo: Imprimir una matriz 5*6 y rellenarla de asteriscos.

```
  entero fila, col
  caracter c = '*'
  Imprimir("imprimir matriz  ",k)
  Para fila = 1, 5,1
       Para col= 1,6, 1
          Imprimir(c)
       Fin Para
     Imprimir( )
  Fin Para
```