UNIDAD CURRICULAR: ALGORITMICA Y PROGRAMACIÓN

UNIDAD V. PROGRAMACIÓN ESTRUCTURADA

CONTENIDO:

UNIDAD V

PROGRAMACIÓN ESTRUCTURADA

La programación estructurada es un conjunto de técnicas que aumentan la productividad del programa reduciendo el elevado tiempo requerido para escribir, verificar, depurar y mantener los programas. La programación estructurada hace los programas más fáciles de escribir, verificar, leer y mantener.

TEOREMA DE LA PROGRAMACIÓN ESTRUCTURADA

En mayo de 1966 Böhm y Jacopini demostraron que un programa propio puede ser escrito utilizando solamente tres tipos de estructura de control:

- Secuenciales
- Selectivas
- Repetitivas

Un programa se define como propio cuando cumple con las siguientes características:

- Posee un solo punto de entrada y uno de salida o fin para control del programa.
- Existen caminos desde la entrada hasta la salida que se pueden seguir y que pasan por todas las partes del programa.
- Todas las instrucciones son ejecutables y no existen lazos o bucles infinitos.

ESTRUCTURAS DE CONTROL

ESTRUCTURA SECUENCIAL

Es aquella en la que una acción (instrucción) sigue a otra en secuencia. Los ejemplos de algoritmos realizados anteriormente tienen estructura secuencial.

ESTRUCTURAS DE DECISIÓN O SELECTIVAS

Se utiliza para tomar decisiones lógicas; de ahí que se suelen denominar estructuras de decisión. Una estructura de decisión dirige el flujo de un programa en una cierta dirección, de entre dos o más posibles, en función de un valor booleano (verdadero o falso). Para ello se evalúa una condición y en función del resultado de la misma se realiza una opción u otra. Las condiciones se especifican usando expresiones lógicas.

Tipos de estructuras de decisión:

- Simple
- Doble o compuesta
- Múltiple
- Anidadas

Estructuras de decisión simple: Ejecuta una determinada acción cuando se cumple una determinada condición.

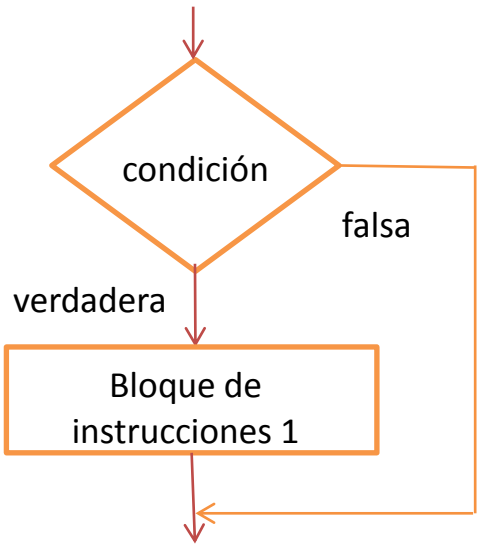
La representación de la estructura de decisión simple en pseudocódigo es:

SI condición ENTONCES

Bloque de instrucciones 1

FIN DEL SI

La representación del diagrama de flujo de una estructura de decisión simple es la siguiente:



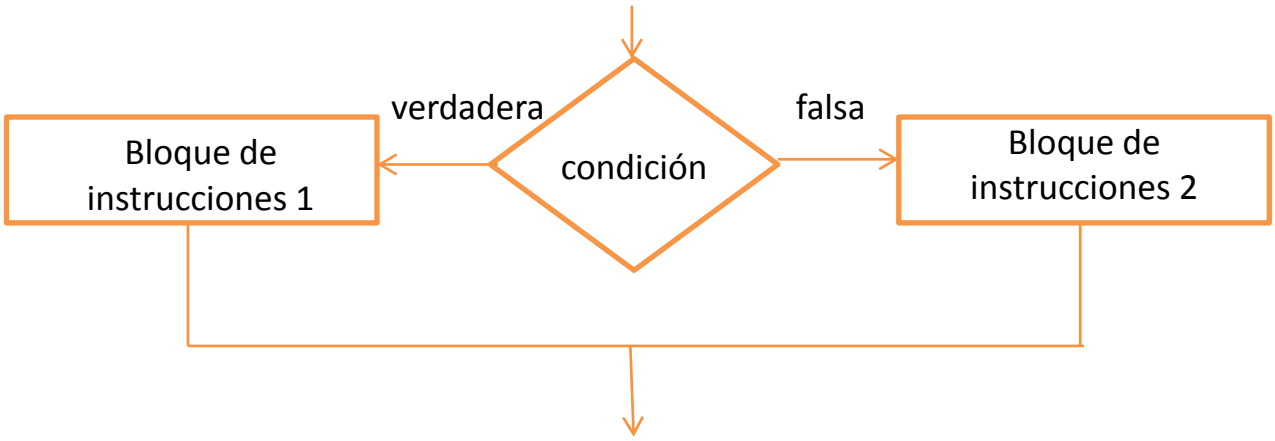
Como puede observarse, al evaluar la condición, si ésta es verdadera se ejecuta el bloque de instrucciones correspondiente. Si la condición da como resultado un valor falso, no se ejecutan esas acciones.
La representación de la estructura de decisión simple en C++ es:

```
if (condición)
{ bloque de instrucciones 1;
}
```

Estructura de decisión doble o compuesta: Se utiliza cuando se requiera elegir entre dos opciones o alternativas posibles, en función del cumplimiento o no de una determinada condición.
La representación de la estructura de decisión doble en pseudocódigo es:

```
SI condición ENTONCES
        Bloque de instrucciones 1
DE LO CONTRARIO
        Bloque de instrucciones 2
FIN DEL SI
```

Nótese que los bloques de instrucciones están indentadas (con sangría), de esta manera aumenta la legibilidad de la estructura.
La representación del diagrama de flujo de una estructura de decisión doble o compuesta es la siguiente:



Como se observa, al evaluar la condición, si ésta es verdadera se ejecuta el bloque de instrucciones 1, pero si la condición da como resultado un valor falso, se ejecuta el bloque de instrucciones 2.

La representación de la estructura de decisión doble en C++ es:

```
if (condición)
{ bloque de instrucciones 1;
}
else
{ bloque de instrucciones 2;
}
```

Estructura de decisión múltiple: Se utiliza cuando se requiera evaluar una expresión que puede tomar varios valores distintos y dependiendo de cada uno de estos valores se ejecutan las acciones.

La representación de la estructura de decisión múltiple en pseudocódigo es:

En caso de *expresión* hacer

Valor1: *bloque de instrucciones 1*

Valor2: *bloque de instrucciones 2*

Valor3: *bloque de instrucciones 3*

.

.

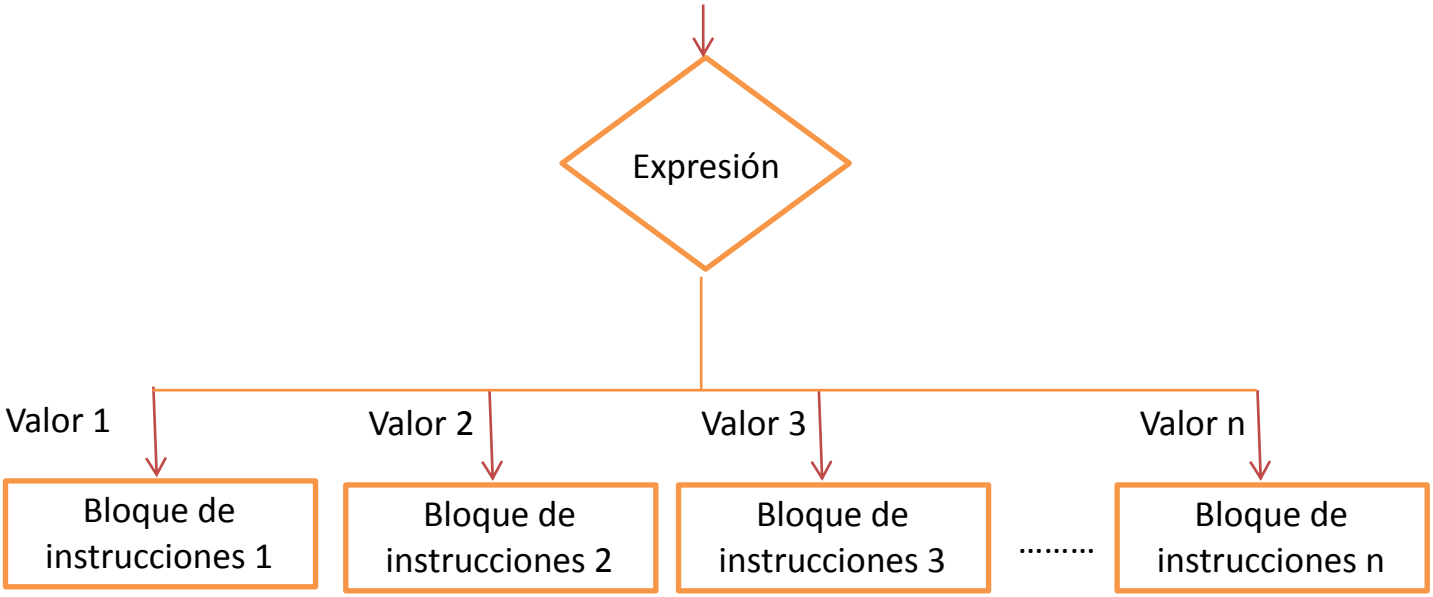
.

.

De lo contrario *bloque de instrucciones n*

Fin_caso

La representación del diagrama de flujo de una estructura de decisión múltiple es la siguiente:



La representación de la estructura de decisión múltiple en C++ es:

```
switch( variable entera, char o booleana)
{
case  valor1: bloque de instrucciones 1;
              break;
case  valor2:  bloque de instrucciones 2;
              break;
 case  valor3: bloque de instrucciones 3;
              break;
   .
   .
   .
default: bloque de instrucciones ;
}
```

Estructura de decisión anidada: Se utiliza cuando se requiera elegir entre mas de dos opciones o alternativas posibles, en función del cumplimiento o no de una determinada condición. Una estructura si-entonces puede contener otra estructura si-entonces, y esta estructura a su vez puede contener otra estructura, y así sucesivamente cualquier número de veces.

La representación de la estructura de decisión anidada en pseudocódigo es:

```
SI condición1 ENTONCES
    SI condición2 ENTONCES
        Bloque de instrucciones 1
    FIN DEL SI
FIN DEL SI
```

Cada condición puede o no tener ***DE LO CONTRARIO***

Una estructura de decisión selectiva también puede construirse con estructuras de decisión anidadas de la siguiente manera:

```
SI condición1 ENTONCES
        Bloque de instrucciones 1
DE LO CONTRARIO
   SI condición2 ENTONCES
        Bloque de instrucciones 2
    DE LO CONTRARIO
        SI condición3 ENTONCES
             Bloque de instrucciones 3
        DE LO CONTRARIO
            .
            .
            .
        FIN DEL SI
    FIN DEL SI
FIN DEL SI
```

La representación de la estructura de decisión anidada en C++ es:

```
if (condición1)
{ bloque de instrucciones 1;
}
else
   if (condición2)
      { bloque de instrucciones 2;
       }
```

```
else
    if (condición3)
      { bloque de instrucciones 3;
       }
    else
      { bloque de instrucciones 2;
       }
```

Ahora bien, antes de profundizar en el uso de cada estructura, expliquemos como se crea una condición.

Condición:

Comúnmente, en una estructura de decisión, la condición es una expresión relacional.

Una condición tiene que ver directamente con una pregunta. La pregunta se forma mínimo con dos operandos y un operador de relación. Cada operando en una expresión relacional puede ser una variable o una constante.

Ejemplos:

a) a>b
b) b>c
c) a=4
d) monto<>c
e) pago<=2000
f) monto>=2200
g) edocivil=”CASADO”

Hay que hacer notar que en C++ algunos operadores cambian con respecto a los que se usan en el pseudocódigo:

Pseudocódigo	C++	Ejemplos en C++
=	==	area == 100
< >	!=	numero != 0
>	>	radio > 1
<	<	edad < 30
>=	>=	temp >= 98.6
<=	<=	tax <= 6.6

Operadores Lógicos

Como se mencionó en la unidad III, los operadores lógicos se utilizan para crear condiciones o expresiones relacionales complejas. Los operadores lógicos son **AND, OR y NOT** y estos se representan en C++ con los símbolos **&&**, **||** y **!**, respectivamente.

Ejemplos de expresiones lógicas:

Pseudocódigo	C++
(voltaje > 48) AND (milliamp < 10)	(voltaje > 48) && (milliamp < 10)
i = j OR a < b	(i == j) || (a < b)
sueldo >= 2000 and numhijos < > 0 or edocivil = 'S'	sueldo >= 2000 && numhijos != 0 || edocivil == 'S'

Ejemplos de expresiones cotidianas convertidas en condiciones (pseudocódigo):

a) Si el empleado tiene menos de 5 años de servicio entonces

Si anoservi<5 entonces

b) Si el trabajador es obrero entonces…

Si tipotrabajador=”OBRERO” entonces

c) Si es una mujer entonces..

Si sexo=”F” entonces…

d) Si el paciente pesa mas de 50 kilos entonces…

Si peso>50 entonces

e) Si el alumno es mayor de edad entonces

Si edad>=18

f) Si el empleado tiene entre 5 y 10 años de servicio entonces…

Si anoservi>=5 AND anoservi<=10 entonces

g) Si es un hombre con mas de 50 años entonces…

Si sexo=”M” AND edad>50 entonces

h) Si gana menos de 5000 Bs. o tiene 5 hijos entonces…

Si sueldo<5000 OR numhijos=5 entonces

i) Si el sueldo está entre 5000 y 13000 Bs. y es viudo entonces…

Si sueldo>=5000 AND sueldo <=13000 AND edocivil=”viudo” entonces

j) Si no es casado pero tiene hijos entonces

Existen varias formas de plantear esta condición:

Si NOT edocivil=”casado” AND Numhijos >0 entonces

Si edocivil<>”casado” AND Numhijos>=1

Si edocivil<>”casado” AND NOT Numhijos=0

Si NOT edocivil=”casado” AND NOT Numhijos=0

Ahora que se conoce la manera de representar una condición procederemos a resolver algunos ejercicios utilizando las distintas estructuras de decisión.

Ejemplos de estructuras de decisión simple:

a) Una empresa paga a sus trabajadores una bonificación que corresponde a 300 Bs por cada hijo y una bonificación por antigüedad de 800 Bs. a los empleados que tienen mas de 5 años de servicio. Determine el monto total a pagar al trabajador.

Algoritmo para calcular bonificaciones

Inicio

Leer numhijos, anos_servi

Monto=300 * numhijos

Si anos_servi>5 entonces

Monto= Monto + 800

Fin del si

Escribir Monto

Fin

b) Dada las 4 notas de un estudiante, determine el promedio de las mismas. Si el promedio es de 17 puntos o mas, emita un mensaje de Felicitaciones.

Algoritmo para calcular promedio de notas

Inicio

Leer nota1, nota2, nota3, nota4

Promedio= (nota1+ nota2+ nota3+ nota4) / 4

Si Promedio>=17 entonces

Escribir “Felicitaciones”

Fin del si

Escribir Promedio

Fin

c) Un vendedor gana una comisión base del 20% de su venta, sin embargo, si su comisión supera los 5000 Bs. se le hace un descuento de 500 Bs. Determine el monto a pagar al vendedor.

Algoritmo para calcular comisión de un vendedor

Inicio

Leer ventas

Comision=ventas*20/100

Si Comision>5000 entonces

Comision= Comision – 500

Fin del si

Escribir Comision

Fin

Ejemplos de estructuras de decisión doble:

a) Dado dos números diferentes determine cuál es el mayor.

Algoritmo para calcular el mayor de dos números

Inicio

Leer num1,num2

Si num1>num2 entonces

Escribir “El mayor es “,num1

De lo contrario

Escribir “El mayor es “, num2

Fin del si

Fin

b) Una empresa paga a sus trabajadores una bonificación que corresponde a 300 Bs por cada hijo y una bonificación por antigüedad de 800 Bs. a los empleados que tienen mas de 5 años de servicio; y de 500 Bs. a los que tienen 5 años de servicio o menos. Determine el monto total a pagar al trabajador.

Algoritmo para calcular bonificaciones

Inicio

Leer numhijos, anos_servi

Monto=300 * numhijos

Si anos_servi>5 entonces

Monto= Monto + 800

De lo contrario

Monto= Monto + 500

Fin del si

Escribir Monto

Fin

c) Una enfermera además de su sueldo base recibe una bonificación de acuerdo al turno en que trabaja. Si trabaja turno diurno le corresponde una bonificación del 10% de su sueldo. Si trabaja turno nocturno le corresponde una bonificación del 20% de su sueldo. Determine el monto total a pagar a la enfermera.

Algoritmo para calcular el monto a pagar a una enfermera

Inicio

Leer sueldo_base, turno

Si turno='D'

15bonificación= sueldo_base * 10/100

De lo contrario

15bonificación= sueldo_base * 20/100

fin del si

total=sueldo_base + 15bonificación

Escribir "El total a pagar es: ", total

Fin

Ejemplo de estructura de decisión múltiple:

a) Teniendo como dato de entrada el Nº del día de la semana (1 al 7) muestre el día de la semana.

Algoritmo Dia de la semana
Inicio
Leer dia
En caso de (*dia)* hacer
1: *Escribir "Lunes"*
2: *Escribir "Martes"*
3: *Escribir "Miércoles"*
4: Escribir "Jueves"
5: Escribir "Viernes"
6: Escribir "Sábado"
7: Escribir "Domingo"
De lo contrario *"Error, debe ingresar un número del 1 al 7"*
Fin_caso
Fin

ESTRUCTURAS DE CONTROL ITERATIVAS

Las estructuras de control iterativas, también denominadas bucles o ciclos repetitivos se utilizan para repetir un conjunto de instrucciones un determinado número de veces.

Tipos de estructuras repetitivas:

- Estructura desde/para
- Estructura mientras
- Estructura repita

Estructura desde/para: Se utiliza cuando se conoce de antemano el número de veces que se desean ejecutar el conjunto de acciones o instrucciones.

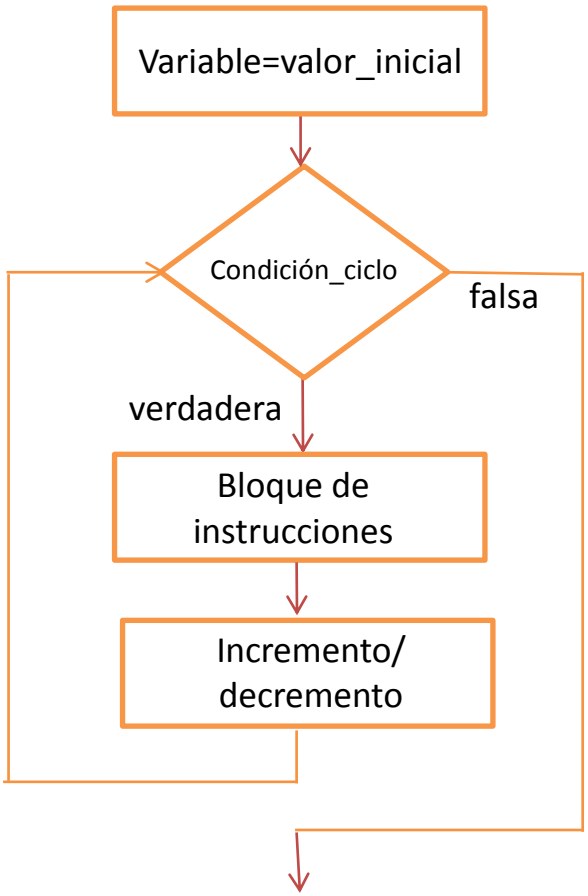
La representación de la estructura ***desde/para*** en pseudocódigo es:

Haga desde variable= *Valor_inicial* **hasta** *Valor_final* (incremento o decremento)

bloque de instrucciones

Fin del haga desde

La representación del diagrama de flujo de una estructura ***desde/para*** es la siguiente:



La representación de la estructura ***desde/para*** en C++ es:

for (variable=*valor_inicial* ; *condición* ; *incremento_o_decremento***)**

{ *bloque de instrucciones;*

}

La variable que se utiliza en la estructura repetitiva se denomina variable de control y se inicializa en un valor. Luego se evalúa la condición. Mientras la condición sea verdadera se repite el bloque el bloque de instrucciones hasta que la condición tome el valor de falsa. La variable se incrementa o decrementa según se indique en la estructura.

Ejemplo:

Mostrar los números del 1 al 10.

Algoritmo para mostrar los números del 1 al 10
Inicio
Haga desde i= 1 hasta 10
    Escribir i
Fin del haga desde
Fin

En c++:

```
for (i=1 ; i<=10; i++)

{ cout << i ;

}
```

En el ejemplo anterior, la variable de control es la variable ***i*** y se inicializa en 1, las instrucciones se repiten mientras el valor de la variable ***i sea menor o igual a 10.*** Con la operación ***i++*** o su equivalente ***i=i+1***, la variable i se incrementa de 1 en 1 (en el algoritmo no es necesario indicar el incremento de 1 en 1). Por lo tanto, se realizarán 10 iteraciones y se generará la siguiente salida: **1 2 3 4 5 6 7 8 9 10.**

Esta estructura repetitiva también puede utilizarse de manera descendente. A continuación se muestra otro ejemplo para mostrar los números del 10 al 1.

Algoritmo para mostrar los números del 10 al 1
Inicio
Haga desde i= 10 hasta 1 i=i-1
    Escribir i
Fin del haga desde
Fin

En c++:

```
for (i=10 ; i>=1; i--)

{ cout << i ;

}
```

Como puede observarse, la variable de control ***i*** se inicializa en 10, las instrucciones se repiten mientras el valor de la variable ***i sea mayor o igual a 1.*** Con la operación ***i - -*** o su equivalente ***i=i-1*** la variable i se decrementa de 1 en 1. Por lo tanto, se realizarán 10 iteraciones y se generará la salida: 10 9 8 7 6 5 4 3 2 1.

Estructura mientras: El ciclo se repite mientras se cumpla una expresión lógica. Por lo tanto, si el valor de la expresión booleana es inicialmente falso, el cuerpo del bucle no se ejecutará.

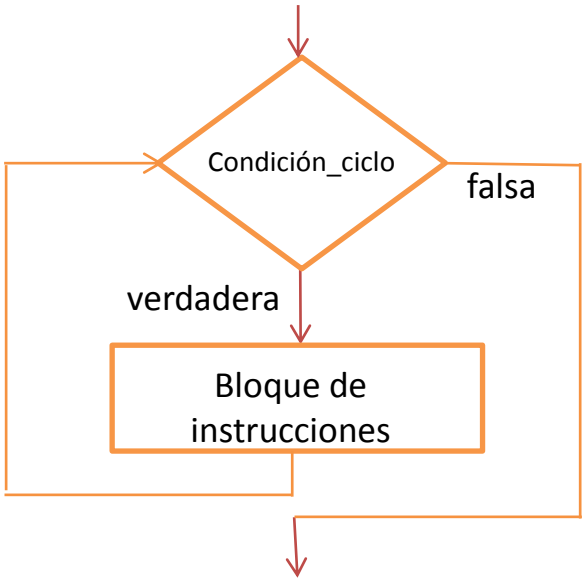
La representación de la estructura ***mientras*** en pseudocódigo es:

Haga mientras *expresión_lógica*

bloque de instrucciones

Fin del haga mientras

La representación del diagrama de flujo de una estructura ***mientras*** es la siguiente:



La representación de la estructura ***mientras*** en C++ es:

```
while (condición_ciclo)
{ bloque de instrucciones;
}
```

Como puede observarse, la estructura repetitiva ***mientras*** tiene una condición o expresión lógica que controla la secuencia de repetición. Esta condición se evalúa antes de que se ejecuten las instrucciones del bucle. Mientras la condición sea verdadera, se repite dicho bloque, hasta que la condición se haga falsa. Por lo tanto, el bloque de instrucciones puede repetirse cero o más veces ya que si inicialmente la condición es falsa, el bloque de instrucciones no se llegará a ejecutar.

Ejemplo:

a) Mostrar los números del 1 al 10.

Algoritmo para mostrar los números del 1 al 10

Inicio

i=1

Haga mientras i<=10

Escribir i

i=i+1

fin del haga mientras

Fin

En c++:

```
i=1;

while (i<=10)
```

```
{ cout << i;

i++;

}
```

b) Mostrar los números del 10 al 1.

Algoritmo para mostrar los números del 10 al 1

```
Inicio

i=10

Haga mientras i>=1

    Escribir i

     i=i-1

fin del haga mientras

Fin
```

En C++:

```
i=10;

while (i>=1)

{ cout << i;

i--;

}
```

Estructura repita: El ciclo se repite al menos una vez antes que se verifique una expresión lógica. Por lo tanto, si el valor de la expresión booleana es inicialmente falso, el cuerpo del bucle se ejecutará al menos una vez.

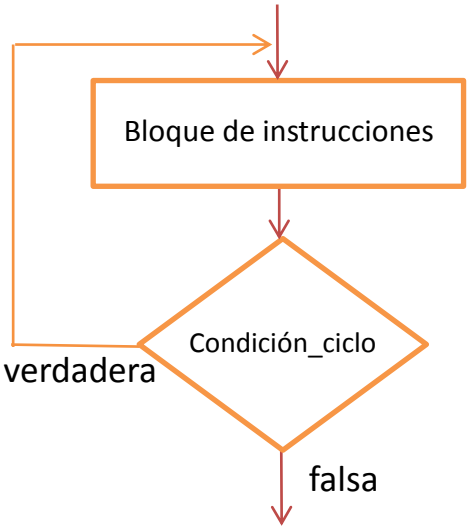
La representación de la estructura ***repita*** en pseudocódigo es:

repetir

bloque de instrucciones

mientras *condición_o_expresión lógica*

La representación del diagrama de flujo de una estructura ***repita*** es la siguiente:



La representación de la estructura ***repita*** en C++ es:

do

{ *bloque de instrucciones;*
}
while (*condición_o_expresión_lógica*);

Ejemplos:

a)Mostrar los números del 1 al 10

Algoritmo para mostrar los números del 1 al 10

Inicio

```
i=1
repita
   Escribir i
    i=i+1
mientras i<=10
Fin
```

En C++:

```
i=1;
do
{
cout<<i;
}
while (i<=10);
```

b) Mostrar los números del 10 al 1.

```
Algoritmo para mostrar los números del 10 al 1
Inicio
i=10
repita
   Escribir i
    i=i-1
mientras i>=1
Fin
```

En C++:

```
i=10;

repita

{ cout << i;

i--;

}
```

```
while (i>=1);
```

Ciclos anidados: Los ciclos o bucles anidados constan de un ciclo externo con uno o más ciclos internos. Cada vez que se repite el bucle externo, los bucles internos se repiten, se reevalúan los componentes de control y se ejecutan todas las iteraciones requeridas.

Ejemplo:

Genere la tabla de multiplicar del 2 al 5

```
Algoritmo tabla de multiplicar del 2 al 5
Inicio
Haga desde i=2 hasta 5
   Haga desde j=1 hasta 10
       producto= i * j
       Escribir i, “x”, j,”=”,producto
   Fin del haga desde
Fin del haga desde
fin
```

EJERCICIOS RESUELTOS

a) Una distribuidora vende desinfectantes a 30 Bs. el litro. Sin embargo, si el cliente compra mas de 12 litros le hace un descuento del 10%. Realice un algoritmo para determinar el monto a pagar.

```
Algoritmo para calcular el monto a pagar por el desinfectante
Inicio
Leer litros
pago=litros*30
Si litros>12
  dcto=pago*10/100
  pago= pago - dcto
fin del si
```

Escribir pago
Fin

b) Para calcular el peso ideal existe una fórmula para los hombres y otra para las mujeres.
Peso ideal de las mujeres= 0,75 (estatura en cms. -150) + 50
Peso ideal de los hombres= 0,75 (estatura en cms. -150) + 55
Realice un algoritmo para determinar el peso ideal de una persona

Algoritmo para calcular el peso ideal de una persona
Inicio
Leer sexo, estatura_cms
Si sexo="F"
    peso_ideal= 0.75 * (estatura_cms - 150) + 50
de lo contrario
    peso_ideal= 0.75 * (estatura_cms - 150) + 55
fin del si
Escribir peso_ideal
Fin

c) El índice de masa corporal (imc) de una persona permite determinar si tiene sobrepeso o desnutrición, y se calcula con la siguiente fórmula:
imc=peso/(estatura en metros^2)
Si el imc es menor 18 hay infrapeso, si el imc oscila entre 18 y 25 el peso es normal y si supera los 25 hay soprepeso. Realice un algoritmo que indique el resultado el imc y su significado.
Algoritmo para calcular el indice de masa corporal de una persona
Inicio
Leer peso, estatura_mts
imc= peso/estatura_mts^2
Escribir imc
Si imc<18
Escribir "Tiene infrapeso"

de lo contrario
  Si imc <=25
    Escribir "Tiene un peso normal"
  de lo contrario
     Escribir "Tiene sobrepeso"
  Fin del si
Fin del si
Fin

d) Realice un algoritmo que calcule la siguiente serie:
1+2+3+4+…..+(n-1)+n
Donde n es un valor dato de entrada.
Algoritmo de cálculo de serie de números
Inicio
Leer n
total=0;
haga desde j=1 hasta n
total=total + j;
fin del haga desde
Escribir total
Fin

e) Mostrar los cuadrados de los números enteros del 1 al 15
Algoritmo para calcular el cuadrado de números
Inicio
haga desde k=1 hasta 15
cuadrado=k * k;
Escribir cuadrado
fin del haga desde
Fin

f)Dadas las notas de 40 estudiantes, calcular el promedio de notas del grupo.
Algoritmo para calcular el promedio de notas

```
Inicio
suma=0
haga desde k=1 hasta 40
Leer nota
suma=suma+nota
fin del haga desde
promedio=suma/40
Escribir promedio
Fin
```

g) Dadas las notas de 40 estudiantes, calcular el número de aprobados y el número de reprobados (se aprueba con 12 puntos)

```
Algoritmo para calcular aprobados y reprobados
Inicio
aprobados=0
reprobados=0
haga desde k=1 hasta 40
Leer nota
Si nota>=12
    aprobados=aprobados+1
de lo contrario
    reprobados=reprobados+1
fin del si
fin del haga desde
Escribir aprobados
Escribir reprobados
Fin
```

REFERENCIAS BIBLIOGRÁFICAS

Bassard, G y Bratley, P. (2010). Fundamentos de algoritmia. Prentice-Hall.

Joyanes, L. (2008). Fundamentos de programación. Algoritmos , Estructuras de datos y objetos. Mc Graw Hill. Tercera edición.

Joyanes, L. y Zahonero, I. (2005). Programación en C. Metodología, algoritmos y Estructura de datos. Mc Graw Hill. Segunda Edición