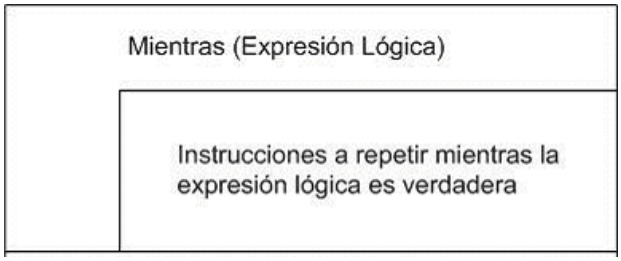
INSTRUCCIÓN WHILE

Una forma más general que el FOR para realizar iteraciones es el WHILE (mientras), el cual permite repetir una serie de instrucciones mientras una determinada expresión lógica sea verdadera. Su forma general utilizando pseudocódigo es la siguiente.

Mientras (expresión lógica)
    Instrucciones a repetir mientras la expresión lógica es verdadera
Fin Mientras

Y en diagrama de caja:



Ejemplo: Diseñar un algoritmo empleando MIENTRAS para escribir 5 veces el texto "Hola" y realizarle prueba de escritorio.

INICIO
    Hacer i = 1
    Mientras (i <= 5)
        Escribir "Hola"
        Hacer i = i + 1
    Fin Mientras
FIN

Nótese que en el MIENTRAS se encuentras las mismas 3 partes que componen el PARA, sólo que no dentro de la misma instrucción.

Como puede observarse en este ejemplo para que el proceso iterativo llevado a cabo en el MIENTRAS no sea infinito, dentro de las instrucciones que se ejecutan en su interior se debe colocar una instrucción que modifique el valor de verdad de la expresión lógica.

Ejemplo: cuantas veces se escribe el texto "Hola" al ejecutar los siguientes algoritmos.

Hacer i = 3
Mientras (i <= 10)
    Escribir "Hola"
Fin Mientras
	Hacer i = 5
	Mientras (i > 0)
	    Escribir "Hola"
	    Hacer i = i + 1
	Fin Mientras
		Hacer i = 2
		Mientras (i < 8)
		    Escribir "Hola"
		    Hacer i = i + 2
		Fin Mientras

De los ejemplos que hemos visto hasta ahora se puede observar que para utilizar un MIENTRAS se deben responder las siguientes preguntas:

En que valor inician la(s) variable(s) de control?
Cual es la condición de terminación?
Como se modifican la(s) variable(s) de control?

Ejemplo: Diseñar un algoritmo para escribir los números impares menores que 100

```
INICIO
        Hacer i = 1
        Mientras (i <= 100)
                Escribir i
                Hacer i = i + 2
        Fin Mientras
FIN
```

Ejemplo: Encontrar el valor de *P* que cumple la siguiente ecuación:

$$\left(\sum_{i=1}^{P} i\right) \leq 1000$$

En otras palabras se pide contar cuantos números enteros positivos se pueden sumar, sin que su suma exceda 1000.

```
INICIO
        Hacer i = 0
        Hacer suma = 0
        Mientras (suma <= 1000)
                Hacer i = i + 1
                Hacer suma = suma + i
        Fin Mientras
        Hacer P = i - 1
        Escribir P
FIN
```

Realizar prueba de escritorio a este algoritmo cambiando 1000 por 10

Ejemplo: Diseñar un programa para escribir los primeros 5 números impares que además sean múltiplos de 3. Realizar prueba de escritorio.

```
INICIO
        Hacer i = 1
        Hacer c = 0
        Mientras (c < 5)
                Si (i % 3 = 0)
```

```
                    Escribir i
                    Hacer c = c + 1
            Fin Si
            Hacer i = i + 2
        Fin Mientras
FIN
```

Importante: Al empezar esta clase se expuso que el WHILE es la forma general del FOR, es decir, que todo algoritmo expresado con FOR se puede "traducir" a WHILE. Sin embargo, en los 2 últimos ejemplos se puede observar que un criterio que puede usarse para saber cuando usar uno u otro es el siguiente: Cuando se requiere realizar un proceso repetitivo y se conoce el número exacto de veces que será llevado a cabo se recomienda usar FOR; por el contrario, si no se sabe cuantas veces se requiere repetir el proceso se recomienda usar WHILE.

El MIENTRAS también puede emplearse para verificar rangos o valores válidos de variables que deban leerse.

Ejemplo: Diseñar un algoritmo para leer una variable X real que representa la nota de un estudiante.

```
INICIO
        Leer X
        Mientras (X < 0 O X > 5)
                Leer X
        Fin Mientras
        Escribir "La nota ha sido leída satisfactoriamente"
FIN
```

Que codificado en lenguaje C queda:

```
#include "iostream.h"

void main()
{
        float X;

        cout<<"Ingrese la nota del estudiante: ";
        cin>>X;

        while (X<0 || X>5)
        {
                cout<<"\n\nValor fuera del rango permitido";
                cout<<"\nIngrese nuevamente la nota: ";
                cin>>X;
        }
```

```
        cout<<"\n\nLa nota ha sido leida satisfactoriamente\n";

}
```

Cómo modificar el algoritmo anterior para que sólo deje realizar 3 intentos para ingresar una nota válida? Codificar este algoritmo en C.

WHILE ANIDADOS

Como hemos visto hasta el momento en el curso, es posible utilizar instrucciones dentro de otras (IF dentro de IF, FOR dentro de FOR, IF dentro de FOR, FOR dentro de IF, etc.). Lo mismo sucede con el WHILE el cual se puede utilizar en combinación bien sea con IF, FOR o con el mismo WHILE.

Ejercicio: Diseñar un algoritmo para escribir los primeros *M* números primos.

Recordemos el algoritmo para determinar si un número era primo o no:

```
INICIO
        Leer X
        Hacer nd = 0
        Para i Desde 1 Hasta X Incrementando de a 1
                Si X % i = 0
                        Hacer nd = nd +1
                Fin Si
        Fin Para
        Si nd > 2
                Escribir "No es primo"
        De lo contrario
                Escribir "Si es primo"
        Fin Si
FIN
```

Cómo modificarlo para que haga lo que necesitamos?

```
INICIO
        Hacer contador = 0
        Hacer X = 1
        Mientras (contador < 100)
                Hacer nd = 0
                Para i Desde 1 Hasta X Incrementando de a 1
                        Si X % i = 0
                                Hacer nd = nd +1
                        Fin Si
                Fin Para
                Si nd <= 2
```

```
                    Hacer contador = contador + 1
                    Escribir contador, X
            Fin Si
            Hacer X = X + 1
        Fin Mientras
FIN
```

Que traducido a C queda:

```
#include "iostream.h"

void main()
{
        int contador, X, nd, i;

        contador = 0;
        X = 1;

        while (contador < 100)
        {
                nd = 0;
                for (i=1; i<= X; i++)
                {
                        if (X%i == 0)
                                nd++;
                }
                if (nd <= 2)
                {
                        contador++;
                        cout<<contador<<":\t"<<X<<"\n";
                }
                X++;
        }

}
```