

Práctica 4: Sentencias de Iteración en C++.

C++ soporta tres tipos de estructuras de iteración: los bucles **while**, **do-while** y **for**. Éstos se corresponden con los esquemas de iteración vistos en el pseudolenguaje definido en Elementos de Programación.

Sentencia **while**

Se corresponde con el esquema **MIENTRAS** del pseudolenguaje. La notación BNF de la sentencia **while** es la siguiente:

```
<sent_while> ::= while '('<expres_log>')' <sent> | <bloque>
```

Un bucle **while** tiene una condición de control o expresión lógica (<expres_log>), que ha de ir encerrada entre paréntesis, que controla la secuencia de repetición. La sintaxis de la sentencia es la siguiente:

```
while (<expres_log>)
{
    <sec_sent>
}
```

Sentencia **do-while**

Se corresponde con el esquema **REPETIR** del pseudolenguaje. La notación BNF de la sentencia **do-while** es la siguiente:

```
<sent_do-while> ::= do <sent>|<bloque> while '('<expres_log>')';
```

En esta sentencia, el cuerpo del bucle se ejecuta hasta que sea **FALSA** la expresión lógica (<expres_log>) (que ha de ir encerrada entre paréntesis). Por tanto, al igual que en el bucle **while** el cuerpo del bucle se ejecuta mientras la expresión lógica sea cierta. Esto supone una diferencia con la sentencia **REPETIR** del pseudolenguaje en la que el cuerpo del bucle se ejecutaba hasta que la expresión lógica fuese verdadera. El bucle **do-while** también se denomina post-prueba, ya que la expresión lógica se comprueba cada vez **después** de la ejecución del cuerpo del bucle. La sintaxis del bucle **do-while** es la siguiente:

```
do
{
    <sec_sent>
}
while (<expres_log>);
```

El cuerpo de una sentencia **do-while** siempre se ejecuta al menos una vez. Cuando esto deba ocurrir en un programa (el cuerpo del bucle tenga que ejecutarse una o más veces), es conveniente el uso de la sentencia **do-while** en lugar de la sentencia **while**.

Sentencia **for**

Se corresponde con el esquema **PARA** del pseudolenguaje. Su notación BNF es la siguiente:

```
<sent_for> ::= for '('<inicializacion> ; <expres_log> ; <incremento> ')'  
              <sent>|<bloque>  
<inicializacion> ::= <ident> = <expresion_constante>
```

donde (<inicializacion>) es la parte en la que se inicializa la variable de control del bucle, (<expres_log>) es una expresión lógica que hace que el bucle ejecute el bloque de sentencias mientras que dicha expresión sea cierta, e (<incremento>) es una expresión que incrementa o decrementa la variable de control del bucle. Una estructura típica **for** sería la siguiente:

```
for (<inicializacion> ; <expres_log> ; <incremento>)  
{  
    <sec_sent>  
}
```

NOTA: NO SE PERMITE EL USO DE LAS SENTENCIAS *break* O *continue* EN UN BUCLE (*for*, *do-while* o *while*) NI LA MODIFICACIÓN DE LA VARIABLE CONTADORA DE UN BUCLE FOR DENTRO DEL MISMO.

ENUNCIADO DE LA PRÁCTICA

1. Escribir un programa que lea 2 números naturales (X e Y) y muestre por pantalla el Máximo Común Divisor y el Mínimo Común Múltiplo de ambos números naturales.

Nota. El MCD de X e Y se calcula por el método de Euclides de forma iterativa mediante las siguientes premisas:

- a) $X > Y$: $\text{MCD}(X, Y) = \text{MCD}(X - Y, Y)$
- b) $\text{MCD}(X, Y) = \text{MCD}(Y, X)$
- c) $\text{MCD}(X, X) = X$

Pensar la relación existente entre los 2 números, su MCD y su MCM.

2. Escribir un programa que presente un menú con las opciones A,B,C,X e imprima por pantalla la opción seleccionada. En el caso de la X deberá preguntarse: “¿Desea salir (S/N)? “ y sólo se saldrá en caso de pulsar la S
3. Escribir un programa que imprima la tabla de multiplicar del 1 al 10) de un número natural leído desde teclado ajustándose al formato del ejemplo.

Ejemplo:

Introduzca un número: 7

La Tabla del 7 es:

=====

```
7 x 1  = 7
7 x 2  = 14
7 x 3  = 21
7 x 4  = 28
7 x 5  = 35
7 x 6  = 42
7 x 7  = 49
7 x 8  = 56
7 x 9  = 63
7 x 10 = 70
```